НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6.050103 “Програмна інженерія’’

на тему: Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка клієнт-серверної архітектури)

Виконав: студент 4 курсу, групи ТІ-51

Бетін Владислав Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. в. Матях Сергій Володимирович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050103 “Програмна інженерія’’

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019 р.

ЗАВДАННЯ
на дипломну роботу студенту
Бетіну Владислав Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи “Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка веб-інтерфейсу користувача)”

керівник роботи ст. в. Матях Сергій Володимирович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від 22 03 2019р. № 1325-с

2. Строк подання студентом роботи ___ _____ 201__ р.

3. Вихідні дані до роботи React.js проект

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення та засоби аналізу сонячної активності, спроектувати архітектуру системи аналізу сонячної активності, розробити програмне забезпечення, розробити інтерфейс користувача

_____
_____
_____
_____

5. Перелік ілюстраційного матеріалу
1. Актуальність 2. Мета та завдання роботи 3. Функції додатку 4. Опис функціональності системи 5. Архітектура програмного комплексу 6. Опис структури інтерфейсу 7.

Дата видачі завдання ”___”_____ 201__ р.

Використати

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2.	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Підготовка матеріалів		
5.	Програмна реалізація системи		
6.	*1* Захист програмного продукту		
7.	Оформлення пояснювальної записки		
8.	*2* Передзахист		
9.	Захист		

при програмні засоби

Студент _____ Бетін В.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Матях С. В.
(підпис) (прізвище та ініціали)

8. Інтерфейс

АНОТАЦІЯ

Метою роботи було створення клієнтського модулю, що дозволяє обраховувати вигідність встановлення сонячних панелей та сонячних колекторів. Система дає можливість користувачу обирати на карті України будь-які точки, для яких проводиться розрахунок видобутку енергії за допомогою сонячної радіації. Застосунок дає можливість вибору різних типів станцій з різними параметрами та введення додаткових параметрів. Додаток представляє собою інтерактивну карту України з даними сонячної інсоляції по регіонам, демонстрацією обчислень видобутку електроенергії для сонячних панелей та гарячої води для геліосистем та відображення результатів у вигляді графіку.

Записка містить 73 сторінки 12 рисунків 6 формул та 9 посилань.

ABSTRACT

The purpose of the work was to create a client module that allows you to calculate the utility of installing solar panels and solar collectors. The system allows the user to select on the map of Ukraine any points for which the calculation of energy production by means of solar radiation is carried out. The application allows the selection of different types of stations with different parameters and the introduction of additional parameters. The annex is an interactive map of Ukraine with solar insolation data from regions, a demonstration of electricity generation calculations for solar panels and hot water for helios systems and displaying results in the form of a graph.

The note contains 73 pages of 12 figures 6 formulas and 9 references.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

БД — база даних;

CRUD (англ. create read update delete) — 4 базові функції управління даними «створення, зчитування, зміна і видалення»;

SQL (англ. Select query language) — декларативна мова програмування для взаємодії користувача з базами даних;

Фреймворк — заготовки, шаблони для програмної платформи, що визначають архітектуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних модулів програмного проекту.

ВСТУП

Ідея отримання енергії з невичерпного джерела завжди була актуальна. Згодом стало з'являтися все більше альтернативних джерел енергії. Вони почали розвиватися і приносити все більше користі. Одним з таких джерел є сонце.

Сонячна енергетика - перспективний напрямок у відновлюваній енергетиці на території України. Важливою перевагою використання енергії сонячної радіації, крім невичерпності потоку її надходження та екологічності, є можливість її використання на віддалених від інфраструктури ділянках земної поверхні з можливістю її безпосереднього перетворення в теплову і електричну енергію.

Раніше обладнання було занадто дорого і його ефективність була досить низькою. Тому альтернативною енергетикою можна було займатися тільки при наявності великих коштів. Але сучасне обладнання набагато ефективніше. Це дозволяє використовувати сонячну енергію не тільки в великих масштабах, але і для побутових цілей. Але перш ніж зважитися на зростання енергії сонця треба проаналізувати безліч факторів, від яких залежить користь, яку отримує споживач.

Тому було запропоновано створити веб-додаток, що буде давати людям можливість отримати приблизні дані того, наскільки вигідно буде встановити той чи інший тип сонячних панелей або геліосистем.

З метою дослідження і розрахунку прибутковості сонячних електростанцій і геліосистем було запропоновано розробити програмну систему, яка продемонструє сонячної активності в регіонах і зможе за індивідуальними даними розрахувати поміс-ячного видобутку електроенергії за допомогою сонячних панелей і теплої води за допомогою геліосистем.

Це рішення допоможе зробити вибір у бік альтернативної енергетики тим людям, які зацікавлені в отриманні енергії від сонце для своїх побутових цілей, але не знають, який тип системи їм потрібен, і як це буде вигідно для їх регіону.

Розроблено додаток при моделюванні сигналу враховує такі особливості, як інтенсивність сонячної інсоляції в залежності від обраного регіону та пори року, кут,

під яким падають сонячні промені, в залежності від пори року, обраний. Дозволяє вибирати різні типи систем перетворення сонячної енергії в електроенергію і теплову енергію. Даний модуль забезпечує можливість проводити розрахунки для отримання інформації у вигляді діаграми.

Записка містить 7 розділів.

У першому розділі описується постановка задачі відображення сонячної активності на території України.

У другому розділі описується характеристика сонячної радіації.

У третьому розділі описані підходи використання сонячної енергії.

У четвертому розділі описані способи аналізу сонячної активності.

У п'ятому розділі вказуються основні засоби розробки даної системи.

У шостому розділі дано опис реалізованого програмного продукту і його архітектури.

У сьомому розділі описано роботу користувача з системою.

ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ ВЕБ-ІНТЕРФЕЙСУ КОРИСТУВАЧА ІНТЕРАКТИВНОЇ ВЕБ-КАРТИ АЛЬТЕРНАТИВНИХ ДЖЕРЕЛ УКРАЇНИ

Мето даної роботи є створення системи для можливості аналізу даних регіонів України по сонячній енергетиці, яка дозволяє аналізувати сонячну інсоляцію на всій території України, передивлятись існуючі станції видобутку енергії з альтернативних джерел, обирати різні типи станцій та проводити розрахунок для кожного типу станції.

З кожним роком сонячна енергетика набуває все більшої популярності. Системи для отримання енергії від сонця з кожним роком стають більш досконалими. Для аналізу вигідності використання цього джерела енергії вже побудовано багато різноманітного програмного забезпечення, яке вирішує дуже широкий спектр прикладних задач. Серед них можна виділити і задачу аналізу сонячної активності в різних регіонах.

З метою дослідження та розрахунку прибутковості сонячних електростанцій і геліосистем було запропоновано розробити програмну систему, яка продемонструє сонячну активність в регіонах та зможе за індивідуальними даними розрахувати помісячного видобутку електроенергії за допомогою сонячних панелей та теплої води за допомогою геліосистем.

Розроблена система повинна забезпечувати наступні можливості:

— Відображення інтенсивності сонячної радіації по регіонам України;

— Створення міток на будь-якій частині карти;

— Можливість розрахунку результатів для сонячних панелей;

— Можливість розрахунку результатів для сонячних колекторів;

— Візуалізація отриманих результатів у формі діаграми;

СОНЯЧНА ЕНЕРГЕТИКА

У даному розділі було проведено аналіз актуальності сонячної енергетики та існуючих програмних рішень.

Потенціал сонячної енергії

Земля отримує 174 петаватт надходить сонячної радіації (інсоляція) у верхній атмосфері. Приблизно 30% відбивається назад у космос, а решту поглинають хмари, океани і суші. Спектр сонячного світла на поверхні Землі в основному поширюється по видимому і ближньому інфрачервоному діапазонах з невеликою частиною в майже ультрафіолетовому світлі. Більшість населення світу живе в районах з рівнем інсоляції 150–300 Вт/м², або 3,5–7,0 кВт-год/м² на добу.

Сонячне випромінювання поглинається земною поверхнею Землі, океанами — які охоплюють близько 71% земної кулі — і атмосферою. Піднімається тепле повітря, що містить випарувану воду з океанів, що викликає циркуляцію повітря або конвекцію. Коли повітря досягає великої висоти, де температура низька, водяна пара конденсується в хмари, що випадають на поверхню Землі, завершуючи кругообіг води. Прихована теплота конденсації води посилює конвекцію, створюючи такі атмосферні явища, як вітер, циклони і антициклони. Сонячне світло, що поглинається океанами і земними масами, зберігає поверхню при середній температурі 14 ° C. За допомогою фотосинтезу зелені рослини перетворюють сонячну енергію в хімічно накопичену енергію, яка виробляє продукти харчування, деревину і біомасу, з якої видобувається викопне паливо.

Загальна сонячна енергія, поглинена атмосферою Землі, океанами і сушами, становить приблизно 3 850 000 екзахоуль (EJ) на рік. У 2002 році це було більше енергії за одну годину, ніж світ, який використовувався протягом одного року. Фотоси-

нтез фіксує приблизно 3000 ЕДж на рік у біомасі. Кількість сонячної енергії, що потрапляє на поверхню планети, настільки велика, що за один рік вона приблизно вдвічі більша, ніж коли-небудь отримана з усіх невідновлюваних ресурсів Землі вугілля, нафти, природного газу та видобутого.

Основними параметрами, які використовуються при визначенні ефективності впровадження сонячних електростанцій, є інтенсивність сонячного випромінювання та зовнішньої температури. Сонячне випромінювання, що надходить на будь-яку поверхню, складається з прямого і дифузного сонячного випромінювання і випромінювання, відбитого від поверхні Землі і різних об'єктів, розташованих поблизу цієї поверхні.

Прямим є потік енергії сонячного випромінювання на одиницю поверхні плоского приймача, розташованого перпендикулярно падаючим променям. Це, відповідно, стосується енергії випромінювання Сонця з нормальним падінням променів. Термін "прямий" вказує, що він відноситься до випромінювання, що передається безпосередньо від Сонця, без будь-якої додаткової кількості розсіяного або відбитого випромінювання, що надходить на Землю після зміни напрямку внаслідок відображення і розсіювання атмосферою.

Поряд з вимірюванням прямого сонячного випромінювання, більш важливим для теоретичних досліджень і практичної реалізації є вимірювання добової сумарної або сумарної сонячної радіації (сума прямого і розсіяного випромінювання) на горизонтальну поверхню.

Кількість енергії сонячного випромінювання значною мірою залежить від астрономічних і метеорологічних факторів — висоти Сонця над горизонтом і довжини дня, хмарності, вологості і прозорості атмосфери.

Надходження від загальної сонячної радіації змінюються протягом дня, року та року за роком, але її середньорічна вартість протягом довгострокового періоду є досить стабільною.

Крім того, самі змінні є складовими загальної сонячної радіації (прямого і дифузного сонячного випромінювання), і часто збільшення одного з значень призводить до зменшення в іншому, майже не впливаючи на їх суму. Пряма складова сонячного

випромінювання в добовій кількості сонячного випромінювання, що падає на горизонтальну поверхню Землі, може бути в межах 90% на дуже ясний день і до 0% у дуже похмурий день. Величина загальної сонячної радіації зі збільшенням висоти рельєфу до 200 м практично не змінюється — у цьому випадку змінюється співвідношення її складових — збільшення величини прямого сонячного випромінювання компенсується зменшенням величини дифузного сонячного випромінювання. Як правило, найбільшу частку в загальній сонячній радіації становить пряме випромінювання (без урахування зимових місяців і окремих районів, наприклад, на півночі).

Величина і співвідношення компонентів загальної сонячної радіації необхідні для вибору типу сонячної енергії обладнання. У регіонах, де переважає пряма сонячна радіація, можна використовувати сонячні колектори з сонячними концентраторами. Дифузне сонячне випромінювання не можна сконцентрувати за допомогою дзеркал; якщо значна частина сонячного випромінювання надходить у вигляді дифузного випромінювання, то використовуються плоскі сонячні колектори, які збирають як пряме, так і дифузне сонячне випромінювання і можуть ефективно використовуватися не тільки на прозорих, але й у похмурі дні.

Потенційна сонячна енергія, яка може бути використана людиною, відрізняється від кількості сонячної енергії, що присутня біля поверхні планети, оскільки такі фактори, як географія, часові відхилення, хмара, і земля, доступна людям, обмежують кількість сонячної енергії, яку ми може придбати.

Географія впливає на потенціал сонячної енергії, оскільки області, які ближче до екватора, мають більшу кількість сонячного випромінювання. Проте використання фотоелектрики, яка може слідувати положенню сонця, може значно збільшити потенціал сонячної енергії в районах, які знаходяться далі від екватора. Зміна часу впливає на потенціал сонячної енергії, тому що в нічний час на поверхні Землі мало сонячного випромінювання для поглинання сонячних панелей. Це обмежує кількість енергії, яку сонячні батареї можуть поглинути за один день. Хмарна оболонка може впливати на потенціал сонячних панелей, оскільки хмари блокують вхідне світло від сонця і зменшують світло, доступне для сонячних батарей.

Крім того, наявність землі має великий вплив на наявну сонячну енергію, оскільки сонячні батареї можуть бути встановлені тільки на землі, яка інакше не використовується і підходить для сонячних панелей. Встановлено, що дахи є придатним місцем для сонячних батарей, оскільки багато людей виявили, що вони можуть збирати енергію безпосередньо зі своїх будинків. Інші області, які підходять для сонячних батарей, - це землі, які не використовуються для підприємств, де можуть бути створені сонячні електростанції .

Сонячні технології можуть бути як пасивні або активні в залежності від способу їх використання, перетворення і розподілу сонячного світла, і дозволяють використовувати сонячну енергію на різних рівнях по всьому світу, в залежності від відстані від екватора. Хоча сонячна енергія в першу чергу стосується використання сонячної радіації для практичних цілей, все поновлювані джерела енергії, крім геотермальної енергії та енергії припливи, отримують свою енергію прямо або побічно від Сонця.

Активні сонячні системи використовують фотоелектрика, теплові сонячні колектори, насоси і вентилятори для перетворення сонячного світла корисними виходи. Пасивні сонячні методи включають вибір матеріалів з сприятливими тепловими властивостями, проектування просторів, які природно циркулюють повітрям, і проектування положення будівлі стороною до сонця. Активні сонячні технології збільшують поставки енергії і вважаються технологіями поставки, тоді як пасивні сонячні технології зменшують потребу в альтернативних ресурсах і, як правило, вважаються технологіями з боку попиту.

Використання сонячної енергії

Сонячна електроенергетика — одна з провідних областей отримання електроенергії за допомогою альтернативних джерел енергії. Сонячна енергія, як очікується, стане найбільшим у світі джерелом електроенергії до 2050 року, а фотоелектрична

енергія і сонячна енергія становлять 16 і 11 відсотків в загальному світовому споживанні, відповідно. У 2016 році, після чергового року швидкого зростання, сонячна енергія створила 1,3% світової енергетики.

Сонячні системи гарячого водопостачання використовують сонячне світло для нагріву води. У низьких географічних широтах (нижче 40 градусів) від 60 до 70% споживання гарячої води для побутових приміщень з температурою до 60 ° С може бути забезпечено сонячними системами опалення. Найбільш поширеними типами сонячних водонагрівачів є евакуйовані трубчасті колектори і глазуровані плоскі колектори, в основному використовуються для гарячої води і неглазуровані пластикові колектори, які використовуються переважно для обігріву басейнів.

Плоскі колектори - найпоширеніші сонячні теплові технології в Європі. Вони складаються з корпусу, що містить темну пластину абсорбера з каналами циркуляції рідини, і прозору кришку, що дозволяє передавати сонячну енергію в корпус. Сторони і задня частина корпусу, як правило, ізольовані, щоб зменшити втрати тепла в навколишньому середовищі. Рідина, що передає тепло, циркулює через канали поглинання рідини для видалення тепла від сонячного колектора. Циркуляційна рідина в тропічному і субтропічному кліматі зазвичай є водою. У кліматі, де, швидше за все, замерзання, замість води або суміші з водою може бути використана рідина для передачі тепла, подібна до автомобільного антифризу. Якщо використовується теплоносійна рідина, зазвичай використовується теплообмінник для передачі тепла від рідини сонячного колектора до резервуара для зберігання гарячої води. Найпоширеніша конструкція абсорбера складається з мідних труб, приєднаних до високопровідного металевого листа (міді або алюмінію). Темне покриття наноситься на сторону обертання, спрямовану до сонця, щоб збільшити його поглинання сонячної енергії. Загальне поглинаюче покриття - чорна емалева фарба.

У конструкціях сонячних колекторів з більш високою продуктивністю прозора кришка має загартоване натрієво-вапняне скло, яке має знижений вміст оксиду заліза, як для фотоелектричних сонячних панелей. Скло може також мати штриховий малюнок і одне або два недзеркальних покриття для подальшого посилення прозорості.

Покриття абсорбера є типово селективним покриттям, де вибіркове значення має спеціальне оптичне властивість для поєднання високого поглинання у видимій частині електромагнітного спектру. Це створює селективну поверхню, що зменшує викид енергії чорного тіла від абсорбера і покращує продуктивність. Трубопроводи можуть бути лазерними або ультразвуковими, привареними до листа абсорбера, щоб зменшити пошкодження селективного покриття, яке зазвичай застосовується до з'єднання з великими котушками в процесі рулону.

Конфігурації поглинаючих трубопроводів включають:

- Арфа — традиційна конструкція з нижніми трубними стояками і верхньою збірною трубою, що використовуються в термосифоні і насосних системах низького тиску;
    - Одна безперервна S-подібна труба, що максимізує температуру, але не загальний вихід енергії в системах змінного потоку, що використовуються в компактних системах побутової гарячої води (відсутність ролі опалення приміщення);
    - Складаються з двох листів металевих формованих для отримання широкої зони циркуляції, що покращує теплообмін;
    - Складається з декількох шарів прозорих і непрозорих листів, які дозволяють поглинати в прикордонному шарі. Оскільки енергія поглинається в прикордонному шарі, перетворення тепла може бути більш ефективним, ніж для колекторів, де поглинене тепло проводиться через матеріал перед тим, як накопичуватися в оборотній рідині.

Комерційно доступний також плоский колектор з використанням стільникової структури для зменшення втрат тепла і на скляній стороні. Більшість плоских колекторів мають тривалість життя більше 25 років.

Багато промислово розвинені країни встановили значну потужність сонячної енергії в свої електромережі, щоб доповнити або забезпечити альтернативу традицій-

ним джерелам енергії, тоді як все більше менш розвинених країн звернулися до сонячної енергії, щоб зменшити залежність від дорогих імпортних палив. Передача на великі відстані дозволяє віддаленим відновлюваним енергоресурсам витісняти споживання викопного палива. Сонячні електростанції використовують одну з двох технологій:

- Фотоелектричні (PV) системи використовують сонячні батареї або на дахах, або на наземних сонячних фермах, перетворюючи сонячне світло безпосередньо в електричну енергію.

- Концентрована сонячна енергія (CSP, також відома як "концентровані сонячні теплові") установки використовують сонячну теплову енергію для виробництва пари, що потім перетворюється в електрику турбіною.

Існуючі рішення

У процесі пошуку інформації, та аналізу існуючих рішень, було виявлено, що на даний момент подібні системи є досить складними у використанні, або реалізують поставлену задачу не в повному обсязі:

— “solar-battery.com.ua” — це веб-сайт, який надає дані сонячної активності по різним областям України. Недоліком даної системи перед поставленою задачею є те, що вона не має опції розрахунку прибутку від сонячних систем та геліосистем. Крім того карта не є інтерактивною, тобто користувач не може з нею взаємодіяти;

— “atmosfera.ua” — це веб-сайт, в якому є калькулятор сонячної електростанції, в якому можна обрати регіон країни та потужність електростанції. Недоліком є те, що неможна вибирати тип сонячних панелей і не зрозуміло для якого типу проводиться обчислення. Також на сайті нема можливості зробити обчислення для геліосистем.

— “enertime.com.ua” — це веб-сайт, в якому є калькулятор сонячної електростанції, в якому можна обрати потужність та тип електростанції, але не можна вибрати регіон країни.

Не у всіх вже існуючих систем є можливість побачити дані по видобутку електроенергії помісячно і не завжди в зручному форматі. Вище приведені сайти використовують лише сонячні батареї і не дають змогу зробити розрахунок для геліосистем. Багато цих рішень є корпоративними, які не дають великий вибір різних типів станцій, але використовують лише свої продукти для розрахунку.

Висновки до розділу

У даному розділі було розглянуто актуальність сонячної енергетики. Було досліджено, що від сонця ми отримуємо дуже велику кількість енергії, яку можна перетворювати як в теплову енергію, так і в електроенергію. Було зроблено аналіз того, які типи систем використовуються для перетворення енергії та які їх характеристики.

СИСТЕМИ ПЕРЕТВОРЕННЯ СОНЯЧНОЇ ЕНЕРГІЇ

Для реалізації програмного продукту треба розуміти, які є різні системи перетворення сонячної енергії. Потрібно знати основні типи цих систем та які характеристики вони мають. Для цього в цьому розділі було розглянуто типи сонячних панелей та сонячних колекторів.

Види колекторів

Сонячний колектор — це пристрій, який збирає та концентрує сонячне випромінювання від Сонця. Ці пристрої використовуються в основному для активного сонячного нагрівання і дозволяють нагрівати воду для особистого користування. Ці колектори, як правило, встановлюються на дах і повинні бути дуже міцними, оскільки вони піддаються різним погодним умовам.

Використання цих сонячних колекторів забезпечує альтернативу традиційному нагріванню води з використанням нагрівача води, потенційно знижуючи витрати енергії з плином часу. Як і в побутових умовах, велика кількість цих колекторів може бути об'єднана в масив і використовується для виробництва електроенергії на сонячних теплових електростанціях.

Є багато різних типів сонячних колекторів, але всі вони побудовані з однаковою базовою передумовою. Загалом, є деякий матеріал, який використовується для збору і фокусування енергії від Сонця і використання його для нагрівання води. Найпростіший з цих пристроїв використовує чорний матеріал, що оточує труби, через які протікає вода. Чорний матеріал дуже добре поглинає сонячне випромінювання, і як матеріал нагріває воду, яку вона оточує. Це дуже простий дизайн, але колектори можуть стати дуже складними. Амортизаторні пластини можна використовувати, якщо підвищення температури не є необхідним, але, як правило, пристрої, які використовують

відбивні матеріали для фокусування сонячного світла, призводять до більшого підвищення температури.

Ці колектори — це просто металеві коробки, які мають якесь прозоре скління, як покриття поверх темно-пофарбованої пластини. Сторони і днище колектора зазвичай покриваються ізоляцією, щоб мінімізувати втрати тепла в інших частинах колектора. Сонячне випромінювання проходить через прозорий склопакет і потрапляє в поглинаючу пластину. Ця пластина нагрівається, передаючи тепло або воді, або повітрю, який утримується між склінням і пластиною абсорбера. Іноді ці абсорберні плити пофарбовані спеціальними покриттями, призначеними для поглинання і утримання тепла краще, ніж традиційна чорна фарба. Ці пластини, як правило, зроблені з металу, який є хорошим провідником — зазвичай міді або алюмінію.

Підрахунок гарячої води сонячних колекторів

Приведена інтенсивність поглинання сонячної радіації ($q_{\theta i,}$ Вт/м^2) для геліоприймача дорівнює:

$$q_{\theta i} = 0{,}96\ (P_s\ \theta_s\ I_s + P_D\ \theta_D\ I_D), \qquad (3.1)$$

де I_S, I_D − інтенсивність сонячного випромінювання, прямого та розсіяного, відповідно, на горизонтальну поверхню, Вт/м^2;

θ_S,θ_D − приведені оптичні характеристики для прямої і розсіяної сонячної радіації відповідно. При відсутності паспортних даних θ_S приймається 0,74, θ_D = 0,64 для геліоприймача з одинарним склом θ_S = 0,63, θ_D = 0,42 для геліоприймача з подвійним склом;

P_S,P_D − коефіцієнти положення сонячного колектора для прямої і розсіяної сонячної радіації відповідно.

Середньомісячні значення коефіцієнта положення сонячного колектора для прямої сонячної радіації P_S для сонячних колекторів південної орієнтації при різних

кутах нахилу їх до горизонту для всіх кліматичних зон України визначаються з табличних даних.

Розрахунковий коефіцієнт положення сонячного колектора для розсіяної сонячної радіації (P_D) визначається:

$$P_D = \cos^2 b/2, \quad (3.2)$$

де b − кут нахилу сонячного колектора до горизонту.

Кут нахилу сонячних колекторів до горизонту для геліоустановок, що працюють круглий рік, приймається рівним величині широти місцевості; при роботі в літній період величині широти місцевості мінус 15°; при роботі в опалювальний період − величині широти місцевості плюс 15°.

Приведена добова інтенсивність поглинання сонячної енергії поверхнею геліоприймача (q, Вт.год) становить:

$$q = F \sum q_{\theta i}, \quad (3.3)$$

де F − площа робочої поверхні геліоприймача, м^2.

Фактична середньодобова кількість теплової енергії ($q_д$,кВт.год), яка поступає до споживача визначається відповідно технічних характеристик вибраного геліотехнічного пристрою (в середньому к.к.д. вітчизняних сонячних колекторів становить 0,5-0,7).

$$q_д = q \times \eta, \quad (3.4)$$

де η −коефіцієнт корисної дії геліотехнічного пристрою.

Середньомісячна ($q_м$) кількість теплової енергії, виробленої геліотехнічним пристроєм визначається як добуток $q_д$ на відповідну кількість робочих днів.

Середньорічна ($q_р$) кількість теплової енергії, виробленої геліотехнічними пристроями визначається сумою середньомісячної кількості теплової енергії, виробленої на протязі року.

Середньодобовий об'єм виробленої геліоустановкою гарячої води ($G_д$, л) визначається за формулою:

$$G_д = F\ \eta \sum q_i/[C_p(t_{вих.} - t_{х.в.})], \quad (3.5)$$

де F − площа геліоприймача, м^2; η − к.к.д. геліоприймача; $\sum q_i$ − кількість теплової енергії, що поглинається поверхнею сонячного колектора, Вт/м^2; C_p − теплоємність води, 1,16 Вт.год/кГ ^oC; $t_{вих.}$ − температура на виході СК, ^oC; $t_{х.в}$ −температура холодної води на вході СК, ^oC.

Середньомісячна ($G_м$) кількість гарячої води, виробленої геліотехнічним пристроєм, визначається як добуток $G_д$ на відповідну кількість днів в місяці; Середньорічна ($G_р$) кількість гарячої води, виробленої геліотехнічними пристроями визначається сумою середньомісячної кількості гарячої води, виробленої на протязі року.

Сонячні батареї

Сонячні батареї використовують сонячне світло як джерело енергії для вироблення електроенергії. Фотоелектричний (PV) модуль являє собою набір елементів, з’єднаних між собою,, зазвичай складається з 6-10 фотоелектричних сонячних елементів. Фотоелектричні модулі складають фотоелектричні систему фотоелектричної системи, яка генерує і постачає сонячна електрика в комерційних і житлових приміщеннях.

Фотоелектричні модулі використовують світлову енергію від сонця, щоб генерувати електрику за допомогою фотоелектричного ефекту. Більшість модулів використовують осередки на основі кристалічного кремнію на основі пластин або тонкоплівкові осередки. Структурний елемент модуля може бути верхнім або заднім шаром. Клітини також повинні бути захищені від механічних пошкоджень і вологи. Більшість модулів жорсткі, але також доступні напівгнучкі, засновані на тонкоплівкових елементах. Елементи повинні бути з'єднані електрично послідовно, одна до іншої.

Деякі спеціальні сонячні фотоелектричні модулі включають концентратори, в яких світло фокусується лінзами або дзеркалами на менші осередки. Це дозволяє використовувати осередки з високою вартістю на одиницю площі (наприклад, арсенід галію) економічно ефективним способом.

Кожен модуль має номінальну вихідну потужність постійного струму яка зазвичай становить від 100 до 365 Вт (Вт). Ефективність модуля визначає площа модуля при тій же номінальній потужності - модуль 230 Вт з ефективністю 8% буде мати подвоєну площу модуля 230 Вт з ККД 16%. Є кілька комерційно доступних сонячних модулів, ефективність яких перевищує 24%.

Залежно від конструкції, фотоелектричні модулі можуть виробляти електрику з діапазону частот світла, але зазвичай не можуть охоплювати весь сонячний спектр (зокрема, ультрафіолетовий, інфрачервоний і слабкий або розсіяне світло). Отже, велика частина енергії падаючого сонячного світла втрачається сонячними модулями.

Один сонячний модуль може виробляти лише обмежену кількість енергії; більшість установок містять кілька модулів. Фотоелектрична система зазвичай включає в себе масив фотоелектричних модулів, інвертор, акумулятор для зберігання, з'єднувальні проводку і, необов'язково, механізм відстеження сонячної енергії.

Підрахунок електроенергії сонячних батарей

При виборі типу та потужності сонячної енергетичної установки для певної місцевості в першу чергу необхідно орієнтуватись на питомі показники з надходження сонячної радіації в даній місцевості (середньомісячна і середньорічна кількість прямої, розсіяної та сумарної сонячної радіації). Дані щодо надходження сонячної енергії в різних областях України можна отримати з даних метеостанцій .

Кількість електроенергії, виробленої сонячними фотоперетворювачами визначається за кількістю сонячної радіації, що поступає на робочу поверхню сонячного фотоперетворювача з врахуванням к.к.д. елемента і дорівнює:

$$E_{кор.} = q_{\theta i} \times \eta, \quad (3.6)$$

де $E_{кор.}$ − корисна електроенергія, тобто та, що поступає до споживача, кВт.год; $q_{\theta i}$ − приведена інтенсивність поглинання сонячної радіації робочою поверхнею сонячного фотоперетворювача, кВт.год; η − к.к.д. сонячного фотоперетворювача.

Висновки до розділу

В даному розділі були розглянуто відмінності від різних типів станцій перетворення сонячної енергії. Було досліджено алгоритми підрахунку електроенергії та гарячої води, отриманих від сонячних панелей та геліостанцій відповідно, та обрано потрібні формули, які будуть використовуватись у веб-карті.

ЗАСОБИ РОЗРОБКИ

Важливо при розробці обирати правильні та зручні інструменти. Одні з таких інструментів є середовище розробки та бібліотеки. Основним середовищем було обрано Visual Studio Code.

Для створення структури інтерфейсу користувача використовувався фреймворк React с мовою програмування TypeScript.

Для підключення інтерактивної карти було обрано бібліотеку Leaflet.js та відкритий провайдер карт OpenStreetMap.

Середовище розробки Visual Studio Code

Visual Studio Code - це редактор вихідного коду, розроблений Microsoft для Windows, Linux і macOS. Він включає підтримку налагодження, вбудований елемент управління Git і GitHub, підсвічування синтаксису, інтелектуальне завершення коду, фрагменти коду і рефакторинг коду. Він легко налаштовується, дозволяючи користувачам змінювати тему, поєднання клавіш, налаштування і встановлювати розширення, які додають додаткові функції. Вихідний код є безкоштовним і відкритим вихідним кодом і випущений під роздільною ліцензією MIT. Скомпільовані виконавчі файли є безкоштовними і безкоштовними для приватного або комерційного використання.

Visual Studio Code заснований на Electron, фреймворку, який використовується для розгортання додатків Node.js для робочого столу, що працює на движку макетів Blink. Хоча він використовує платформу Electron, програмне забезпечення не використовує Atom і замість цього використовує той же компонент редактора (під кодовою назвою «Monaco»), який використовується в DevOps Azure (раніше називався Visual Studio Online і Visual Studio Team Services).

В опитуванні розробників Stack Overflow 2019 код Visual Studio був визнаний найпопулярнішим інструментом середовища розробки, причому 50,7% з 87 317 респондентів заявили, що його використовують.

Фреймворк React

React (також відомий як React.js або ReactJS) - це бібліотека JavaScript для створення користувацьких інтерфейсів. Він підтримується Facebook і співтовариством окремих розробників і компаній.

React можна використовувати в якості основи при розробці односторінкових або мобільних додатків, оскільки він оптимальний для вилучення швидко мінливих даних, які необхідно записати. Однак вибірка даних - це тільки початок того, що відбувається на веб-сторінці, тому складні додатки React зазвичай вимагають використання додаткових бібліотек для управління станом, маршрутизації і взаємодії з API.

В основі всіх реактивних додатків лежать компоненти. Компонент - це самостійний модуль, який видає деякий вивід. Ми можемо писати елементи інтерфейсу, як кнопка або поле введення, як компонент React. Компоненти складаються. Компонент може включати в себе один або більше інших компонентів.

У широкому розумінні, для написання React apps пишемо React-компоненти, які відповідають різним елементам інтерфейсу. Потім ми організуємо ці компоненти всередині компонентів вищого рівня, які визначають структуру нашої програми.

Наприклад, візьміть форму. Форма може складатися з багатьох елементів інтерфейсу, таких як поля введення, мітки або кнопки. Кожен елемент всередині форми може бути записаний як компонент React. Потім ми запишемо компонент вищого рівня, сам компонент форми. Компонент форми буде визначати структуру форми і включати в себе кожен з цих елементів інтерфейсу.

Важливо відзначити, що кожен компонент програми React дотримується принципів жорсткого управління даними. Складні, інтерактивні інтерфейси користувача часто включають складні дані і стан програми. Площа поверхні React є обмеженою і

спрямована на те, щоб дати нам інструменти, щоб ми могли передбачити, як наша програма буде виглядати з заданим набором обставин. Ми заглиблюємося в ці принципи пізніше.

На відміну від багатьох попередників, React працює не безпосередньо на об'єктній моделі документів браузера (DOM), а на віртуальному DOM. Тобто, замість того, щоб маніпулювати документом у браузері після зміни наших даних (які можуть бути досить повільними), він вирішує зміни на DOM, побудованому і запущеному повністю в пам'яті. Після оновлення віртуального DOM, React розумно визначає, які зміни потрібно внести до фактичного DOM браузера.

React Virtual DOM цілком існує в пам'яті і є поданням DOM веб-браузера. Через це, коли ми пишемо компонент React, ми не пишемо безпосередньо до DOM, але ми пишемо віртуальний компонент, який React перетворить на DOM.

React використовуэться для побудови Single page application веб-сайтів.

Односторінкове додаток (SPA) - це веб-додаток або веб-сайт, який взаємодіє з користувачем, динамічно переписуючи поточну сторінку, а не завантажуючи всі нові сторінки з сервера. Цей підхід дозволяє уникнути переривання користувацького досвіду між послідовними сторінками, роблячи програму більш схожою на настільну програму. У SPA, або весь необхідний код - HTML, JavaScript, і CSS - витягується з однієї завантаження сторінки, або відповідних ресурсів динамічно завантажуються і додаються на сторінку, як це необхідно, як правило, у відповідь на дії користувача. Сторінка не перезавантажується в будь-який момент процесу, а також не здійснює керування передачею на іншу сторінку, хоча хеш розташування або API історії HTML5 можна використовувати для забезпечення сприйняття та навігації окремих логічних сторінок програми. Взаємодія з програмою для однієї сторінки часто передбачає динамічну комунікацію з веб-сервером за кадром.

Оскільки SPA - це еволюція, яка відійшла від моделі безсторінкового перемальовування сторінок, на яку спочатку були розроблені браузери, з'явилися нові виклики. Кожна з цих проблем має ефективне рішення:

- Бібліотеки JavaScript на стороні клієнта, що вирішують різні проблеми.

- Серверні веб-фреймворки, які спеціалізуються на моделі SPA.
- Еволюція браузерів і специфікація HTML5, розроблена для моделі SPA.

Через відсутність виконання JavaScript на сканерах деяких популярних веб-пошукових систем, SEO (Search engine optimization) історично представляла проблему для публічних веб-сайтів, які бажають прийняти модель SPA.

У період між 2009 і 2015 роками Google Webmaster Central запропонував, а потім рекомендував "схему сканування AJAX", використовуючи початковий знак оклику у ідентифікаторах фрагментів для AJAX-сторінок. Спеціальна поведінка повинна бути реалізована на сайті СПА, щоб дозволити вилучення відповідних метаданих сканером пошукової системи. Для пошукових систем, які не підтримують цю схему хешування URL, хешировані URL-адреси SPA залишаються невидимими. Ці "хеш-банг" URI були розглянуті проблематично багатьма письменниками, включаючи Jeni Tennison на W3C, оскільки вони роблять сторінки недоступними для тих, хто не активував JavaScript у своєму браузері. Вони також розривають заголовки рефереру HTTP, оскільки браузерам не дозволяється відправляти ідентифікатор фрагмента в заголовку Referer. У 2015 році компанія Google знехтувала свою скануючу пропозицію AJAX.

Альтернативно, програми можуть відтворювати завантаження першої сторінки на сервері і наступні оновлення сторінок на клієнті. Це традиційно важко, тому що код рендеринга може знадобитися записати на іншій мові або фреймворку на сервері і в клієнті. Використання шаблонів без логіки, перехресна компіляція з однієї мови на іншу або використання однієї мови на сервері та клієнті може допомогти збільшити кількість коду, який можна спільно використовувати.

Оскільки сумісність SEO не є тривіальною в СПА, варто зазначити, що СЗЗ зазвичай не використовуються в контексті, де індексація пошукових систем є або вимогою, або бажаною. Випадки використання включають програми, які повертають приватні дані, приховані за системою аутентифікації. У тих випадках, коли ці програми є споживчими продуктами, часто використовується класична модель "перемальовування сторінок" для цільової сторінки додатків і маркетингового сайту, що забезпечує

достатню кількість метаданих, щоб програма відображалася як запит у пошуковому запиті. Блоги, форуми підтримки та інші традиційні артефакти перемальовування сторінок часто сидять навколо SPA, що дозволяє насінню пошукових систем з відповідними термінами.

Інший підхід, який використовують серверно-орієнтовані веб-фреймворки, такі як ItsNat, заснований на Java, полягає у відтворенні будь-якого гіпертексту на сервері з використанням тієї ж самої мови і технології шаблонів. У цьому підході сервер з точністю знає стан DOM на клієнті, будь-яке велике або мале оновлення сторінки, яке потрібно, генерується на сервері і транспортується Ajax, точний код JavaScript, щоб привести сторінку клієнта до нового стану, що виконує методи DOM . Розробники можуть вирішити, які стани сторінки повинні бути скановані веб-павуками для SEO, і мати змогу генерувати необхідний стан під час завантаження, генеруючи звичайний HTML замість JavaScript. У разі рамки ItsNat, це автоматично, оскільки ItsNat зберігає дерево DOM клієнта на сервері як дерево Java DOM W3C; рендеринг цього дерева DOM на сервері генерує звичайний HTML під час завантаження і дії JavaScript DOM для запитів Ajax. Ця двоїстість дуже важлива для SEO, тому що розробники можуть створювати з тим же Java-кодом і чистим HTML-шаблоном бажаного стану DOM на сервері; на час завантаження сторінки звичайний HTML генерується за допомогою ItsNat, що робить цей стан DOM сумісним з SEO. З версії 1.3, ItsNat надає новий режим без статусу, а клієнт DOM не зберігається на сервері, тому що з клієнтом режиму без участі, стан DOM частково або повністю реконструюється на сервері при обробці будь-якого запиту Ajax на основі необхідні дані, відправлені клієнтом, інформуючи сервер поточного стану DOM; режим «без статусу» може бути також SEO-сумісним, тому що SEO-сумісність відбувається під час завантаження початкової сторінки, що не залежить від режиму стану або безгромадянства.

Є кілька обхідних шляхів, щоб зробити його схожим на те, що веб-сайт можна сканувати. І те й інше передбачає створення окремих сторінок HTML, які відображають зміст SPA. Сервер може створити HTML-версію сайту і доставити його для сканерів, або можна використовувати безголовий браузер, такий як PhantomJS, для запуску програми JavaScript і виведення результуючого HTML.

Обидва з них вимагають досить багато зусиль, і в кінцевому підсумку може дати головний біль для великих складних об'єктів. Є також потенційні пастки SEO. Якщо сервер, створений HTML, вважається занадто відмінним від вмісту SPA, то сайт буде покараний. Запуск PhantomJS для виведення HTML може уповільнити швидкість реакції сторінок, що є те, для чого пошукові системи — зокрема Google — знижують рейтинг.

Один із способів збільшити обсяг коду, який можна розділити між серверами та клієнтами, — це використання мови шаблонів без логіки, таких як вуса або ручки. Такі шаблони можуть відображатися з різних мов, таких як Ruby на сервері і JavaScript в клієнті. Однак, просто обмін шаблонами зазвичай вимагає дублювання бізнес-логіки, що використовується для вибору правильних шаблонів і заповнення їх даними. Візуалізація з шаблонів може мати негативні ефекти продуктивності лише при оновленні невеликої частини сторінки — наприклад, значення введення тексту в великому шаблоні. Заміна всього шаблону може також порушити вибір користувача або позицію курсора, де оновлення тільки зміненого значення може і не бути. Щоб уникнути цих проблем, програми можуть використовувати прив'язки даних інтерфейсу користувача або гранульовану маніпуляцію DOM, щоб оновлювати лише відповідні частини сторінки замість того, щоб повторно відображати цілі шаблони.

За допомогою SPA, за визначенням, "single page", модель розбиває дизайн браузера для навігації історії сторінок за допомогою кнопок Переслати / Назад. Це створює перешкоди при використанні, коли користувач натискає кнопку "назад", очікуючи попереднього стану екрану в межах SPA, але замість цього, одна сторінка сторінки програми вивантажується, а попередня сторінка в історії браузера представлена.

Традиційним рішенням для SPA є зміна ідентифікатора хеш-фрагмента URL браузера відповідно до поточного стану екрана. Це може бути досягнуто за допомогою JavaScript, і викликає події історії URL в браузері. До тих пір, поки SPA здатна воскресити одне і те ж стан екрана від інформації, що міститься в хеші URL, очікувана поведінка кнопки назад зберігається.

Для подальшого вирішення цієї проблеми, специфікація HTML5 запровадила pushState і замінила державу, яка надає програмний доступ до фактичної URL-адреси та історії браузера.

Фреймворк Leaflet

Leaflet — широко використовувана бібліотека відкритих кодів JavaScript, яка використовується для побудови веб-картографічних додатків. Вперше випущений в 2011 році, він підтримує більшість мобільних і настільних платформ, підтримуючи HTML5 і CSS3. Поряд з OpenLayers і API Карт Google він є однією з найпопулярніших бібліотек для відображення JavaScript і використовується основними веб-сайтами, такими як FourSquare, Pinterest і Flickr.

Leaflet дозволяє розробникам без ГІС-фону дуже легко виводити плиткові веб-карти, розміщені на загальнодоступному сервері, з додатковими плитковими накладками. Він може завантажувати дані про функції з файлів GeoJSON, стилювати їх і створювати інтерактивні шари, такі як маркери з спливаючими вікнами при натисканні.

Leaflet підтримує шари Web-сервісу (WMS), шари GeoJSON, векторні шари та шари плитки. Багато інших типів шарів підтримуються за допомогою плагінів.

Як і інші бібліотеки веб-карт, основна модель відображення, реалізована Leaflet, — це одна базальна карта, плюс нуль або більше прозорих накладок, при цьому на вершині відображаються нуль або більше векторних об'єктів. Елементи

Основними типами об'єктів є: растрові типи (TileLayer і ImageOverlay), векторні типи (Path, Polygon, and specific types such as Circle), мішаніі типи (LayerGroup, FeatureGroup і GeoJSON) та компоненти керування (масштабування, шари тощо).

Існує також цілий ряд утилітних класів, таких як інтерфейси для управління проекціями, перетвореннями та взаємодією з DOM.

Leaflet безпосередньо порівнянна з OpenLayers, оскільки обидві з них є відкритими, клієнтськими лише бібліотеками JavaScript. Бібліотека в цілому набагато менша, близько 7000 рядків коду в порівнянні з 230,000 OpenLayers (станом на 2015 рік). Вона має менший розмір коду, ніж OpenLayers (близько 123 Кб проти 423 Кб), що частково пояснюється його модульною структурою. Кодова база є більш новою і використовує переваги останніх функцій JavaScript, а також HTML5 і CSS3. Тим не менш, Leaflet не вистачає функції OpenLayers підтримує, такі як Web Feature Service (WFS) і вбудована підтримка проекцій, крім Google Web Mercator (EPSG 3857).

Він також можна порівняти з закритим вихідним кодом Google Maps API (дебютував у 2005 році) і Bing Maps API, обидва з яких мають значний серверний компонент для надання таких послуг, як геокодування, маршрутизація, пошук та інтеграція з такими функціями, як Google Google Maps API забезпечує швидкість і простоту, але не є гнучкою і може бути використана лише для доступу до служб Google Maps. Однак нова частина даних DataLayer API Google дозволяє відображати зовнішні джерела даних.

Для роботи з даними якихось регіонів на карті потрібно використовувати формат GeoJson. GeoJSON — це відкритий стандартний формат, розроблений для подання простих географічних об'єктів, а також їх непросторових атрибутів. Він заснований на JSON, нотації об'єктів JavaScript.

Основним елементом географічних даних є координати. Це єдине число, що представляє один вимір: зазвичай розміри - довгота і широта. Іноді існує також координата для висоти. Час — це розмірність, але зазвичай не відображається в координаті, тому що він занадто складний, щоб вписатися в число.

Координати в GeoJSON відформатовані як цифри в JSON: у простому десятковому форматі. На відміну від географічних даних для людського споживання, формати даних ніколи не використовують кодування, не пов'язане з базою, як, наприклад, сексуальне. Настільки ж прохолодно, як і 8 ° 10 '23', це просто не дуже хороший спосіб розкрити номери на комп'ютерах.

Позиція - це масив координат: це найменша одиниця, яку ми дійсно можемо вважати "місцем", оскільки вона може представляти точку на землі. GeoJSON описує замовлення на координати: вони повинні йти у такій послідовності: довгота, широта, висота.

Цей порядок може бути дивним. Історично, порядок координат, як правило, "широта, довгота", і багато людей припускатимуть, що це справа універсально. Протягом довгих годин обговорювалися питання, які краще, але для цього обговорення я підсумую такі:

- Довгота, широта відповідає X, Y порядку математики
- Формати даних зазвичай використовують порядок довготи, широти
- Додатки мають тенденцію до використання широти, довготи

Перш ніж випустити поточну специфікацію, GeoJSON дозволив зберігати більше 3 координат на позицію, а іноді люди використовували це для зберігання часу, частоти серцевих скорочень і так далі. Цей інструмент не підтримується належним чином у інструментах GeoJSON, і його заборонено новою специфікацією.

В плані продуктивності потрібно зразу зрозуміти, де в вашій системі буде вузьке місце. Наприклад, якщо у вас є класичні проблеми з налаштуванням і продуктивністю GeoJSON + Leaflet, вузьким місцем майже завжди є мережа або SVG. Якщо це продуктивність SVG — вартість листівки для малювання полігонів і рядків у вашому веб-переглядачі — тоді формат файлу не має значення. Передайте ті ж самі дані в ультра-ефективному форматі, і ви все одно отримаєте повільну карту.

Припустимо, що мережевий час є вузьким місцем: файл GeoJSON має 20 МБ і завантажується за 20 секунд. Підходи до вирішення такого роду питань є більш загальними, ніж будь-який формат файлів: стиснення з втратами, завантаження підмножин, передавання у вигляді потоку.

Мова TypeScript

TypeScript - це мова програмування з відкритим вихідним кодом, розроблена та підтримувана корпорацією Майкрософт. Це сувора синтаксична надмножина JavaScript і додає до мови додаткову статичну типізацію.

TypeScript призначений для розробки великих додатків і транскомпілюється на JavaScript. Оскільки TypeScript є набором JavaScript, існуючі програми JavaScript також є дійсними програмами TypeScript. TypeScript може використовуватися для розробки програм JavaScript для виконання на стороні клієнта та на стороні сервера (Node.js).

Існує декілька варіантів для транскомпіляції. Можна використовувати за замовчуванням TypeScript Checker, або компілятор Babel можна викликати для перетворення TypeScript у JavaScript.

TypeScript підтримує файли визначення, які можуть містити інформацію про типи існуючих бібліотек JavaScript, подібно до того, як файли заголовків C ++ можуть описувати структуру існуючих файлів об'єктів. Це дозволяє іншим програмам використовувати значення, визначені у файлах, як якщо б вони були статично типізованими сутностями TypeScript. Існують файли заголовків сторонніх виробників для популярних бібліотек, таких як jQuery, MongoDB і D3.js. Доступні також заголовки TypeScript для базових модулів Node.js, що дозволяють розробляти програми Node.js у межах TypeScript.

Компілятор TypeScript сам написаний у TypeScript і компілюється у JavaScript. Він ліцензований під ліцензією Apache 2.0. TypeScript включений як першокласний мова програмування в Microsoft Visual Studio 2013 Update 2 і пізніших версіях, поряд з C # та іншими мовами Microsoft. Офіційне розширення дозволяє Visual Studio 2012 також підтримувати TypeScript.

TypeScript походить від недоліків JavaScript для розробки великомасштабних програм як у Microsoft, так і серед їхніх зовнішніх клієнтів. Проблеми зі складним

кодом JavaScript призвели до потреби в користувальницьких інструментах для полегшення розробки компонентів мовою.

Розробники TypeScript шукали рішення, яке б не порушувало сумісність зі стандартом і його підтримкою між платформами. Знаючи, що поточна стандартна пропозиція ECMAScript обіцяє майбутню підтримку для програмування на основі класів, TypeScript базувався на цій пропозиції. Це призвело до компілятора JavaScript з набором розширень синтаксичних мов, розширених на основі пропозиції, що перетворює розширення у звичайний JavaScript. У цьому сенсі TypeScript був попереднім переглядом того, чого очікувати від ECMAScript 2015. Унікальний аспект, який не входить до пропозиції, але додається до TypeScript, є додатковим статичним типізацією, що дозволяє статичний аналіз мови, що полегшує інструменти та підтримку IDE.

Компілятор TypeScript, названий tsc, написаний у TypeScript. Як наслідок, його можна скомпілювати в звичайний JavaScript, а потім виконати в будь-якому движку JavaScript (наприклад, браузер). Пакет компілятора поставляється в комплекті з хостом сценарію, який може виконувати компілятор. Він також доступний як пакет Node.js, який використовує Node.js як хост.

Існує також альфа-версія компілятора на стороні клієнта в JavaScript, який виконує код TypeScript на льоту, при завантаженні сторінки.

Поточна версія компілятора підтримує ECMAScript 5 за замовчуванням. Дозволяється націлювати ECMAScript 2015 на використання функцій мови, ексклюзивних для цієї версії (наприклад, генераторів). Класи, незважаючи на те, що вони є частиною стандарту ECMAScript 2015, доступні в обох режимах.

TypeScript є надбудовою над ECMAScript 2015 з суворою типізацією, який сам по собі є набором ECMAScript 5, який зазвичай називають JavaScript. Таким чином, програма JavaScript також є дійсною програмою TypeScript, а програма TypeScript може безперешкодно споживати JavaScript. За замовчуванням компілятор націлює ECMAScript 5, поточний стандарт, що переважає, але також може генерувати конструкції, що використовуються в ECMAScript 3 або 2015.

За допомогою TypeScript можна використовувати існуючий JavaScript-код, включати популярні бібліотеки JavaScript і викликати код, створений за допомогою

TypeScript, з іншого JavaScript. Декларації типу для цих бібліотек надаються з вихідним кодом. Анотації для примітивних типів - це числа, булеві та рядки. Слабо- або динамічно-типізовані структури мають будь-який тип.

Типи анотацій можна експортувати в окремий файл декларацій, щоб зробити інформацію про тип доступною для сценаріїв TypeScript, використовуючи типи, вже зібрані в JavaScript. Анотації можуть бути оголошені для існуючої бібліотеки JavaScript, як це було зроблено для Node.js і jQuery.

Компілятор TypeScript використовує виведення типів для виведення типів, коли типи не надаються. Наприклад, метод add в наведеному вище коді буде визначено як повернення числа, навіть якщо не було надано анотацію типу повернення. Це засноване на статичних типах лівих і правих чисел, і знання компілятора, що результат додавання двох чисел завжди є числом. Проте, явне оголошення типу повернення дозволяє компілятору перевірити правильність.

Якщо жоден тип не може бути виведений через відсутність декларацій, то за замовчуванням він буде динамічним будь-якого типу. Значення будь-якого типу підтримує ті ж операції, що і значення JavaScript, а мінімальна статична перевірка типу виконується для операцій з будь-якими значеннями.
Під час компіляції сценарію TypeScript є можливість генерувати файл декларації (з розширенням .d.ts), який функціонує як інтерфейс для компонентів у скомпльованому JavaScript. У процесі роботи компілятор знімає всі тіла функцій і методів і зберігає тільки підписи типів, які експортуються. Отриманий файл декларації може бути використаний для опису експортованих типів віртуального TypeScript бібліотеки або модуля JavaScript, коли сторонній розробник споживає його з TypeScript.

Концепція файлів декларацій аналогічна концепції заголовного файлу, знайденого в C / C ++.

Файли типів оголошень можуть бути написані вручну для існуючих бібліотек JavaScript, як це було зроблено для jQuery і Node.js.
TypeScript розрізняє модулі та простори імен. Обидві функції в TypeScript підтримують інкапсуляцію класів, інтерфейсів, функцій і змінних в контейнери.

Карта OpenStreetMap

OpenStreetMap (OSM) - це спільний проект для створення вільної карти для редагування світу. Дані, згенеровані проектом, замість самої карти вважаються її первинним виходом. Створення та зростання ОСМ мотивувалося обмеженнями використання чи доступності картографічної інформації в більшій частині світу, а також появою недорогих портативних супутникових навігаційних пристроїв. OSM вважається яскравим прикладом добровільної географічної інформації.

Створений Steve Coast у Великобританії в 2004 році, він був натхненний успіхом Вікіпедії і переважанням власних картних даних у Великобританії та інших країнах. З тих пір вона зросла до більш ніж двох мільйонів зареєстрованих користувачів, які можуть збирати дані за допомогою ручного огляду, GPS-пристроїв, аерофотозйомки та інших вільних джерел. Дані дані, надані за посередництвом, передаються за ліцензією Open Database. Сайт підтримується Фондом OpenStreetMap, неприбутковою організацією, зареєстрованою в Англії та Уельсі.

Дані з OSM доступні для використання в обох традиційних додатках, таких як використання їх у Facebook, Craigslist, OsmAnd, Geocaching, MapQuest Open, статистичне програмне забезпечення JMP і Foursquare для заміни Карт Google, а також більш незвичайні ролі, наприклад, замінити дані за умовчанням GPS-приймачі. Дані OpenStreetMap позитивно порівнюються з власними джерелами даних, хоча в 2009 році якість даних відрізнялася в усьому світі

Дані карти зібрані з нуля добровольцями, які здійснюють систематичні обстеження на землі за допомогою таких інструментів, як ручний GPS-пристрій, ноутбук, цифрова камера або диктофон. Дані вводяться в базу даних OpenStreetMap. Марафонові змагання проводяться також командою OpenStreetMap та неприбутковими організаціями та органами місцевого самоврядування для відображення конкретної області.

Наявність аерофотозйомки та інших даних з комерційних та урядових джерел додало важливих джерел даних для ручного редагування та автоматизованого імпорту. Існують спеціальні процеси для автоматичного імпортування та уникнення юридичних та технічних проблем.

Редагування карт можна зробити за допомогою редактора веб-переглядача за замовчуванням, який називається iD, додатком HTML5, який використовує D3.js і написаний Mapbox, який спочатку фінансувався Фондом Найта. Для користувачів середнього рівня зберігається раніше застосована програма Potlatch на основі Flash. JOSM і Merkaartor є більш потужними програмами для редагування на робочому столі, які краще підходять для досвідчених користувачів.

Vespucci - перший повнофункціональний редактор для Android; вона була випущена в 2009 році. StreetComplete - це нова, зручна програма для Android, запущена в 2016 році, яка дозволяє користувачам без будь-яких знань OpenStreetMap відповідати на прості квести для існуючих даних у OpenStreetMap, і таким чином сприяти передачі даних. Maps.me - це мобільний додаток (який працює на обох платформах Android і iOS) і пропонує автономні карти, які також містять обмежений редактор даних OSM. Go Map!! це додаток iOS, що дозволяє створювати та редагувати інформацію в OpenStreetMap. Pushpin - це інша додаток iOS, що дозволяє додавати POI на ходу.

Проект має географічно різноманітну користувацьку базу, завдяки акцентуванню місцевих знань і основних прав у процесі збору даних. Багато ранніх учасників були велосипедистами, які обстежували велосипедистів і переглядали велосипедні маршрути і судна. Інші - професіонали ГІС, які надають дані з інструментами Esri. Співробітниками є переважно чоловіки, причому лише 3–5% складають жінки.

До серпня 2008 року, незабаром після другої конференції «Держава на карті», було зареєстровано понад 50 000 учасників; до березня 2009 року їх було 100 тисяч, а до кінця 2009 року ця цифра становила майже 200 тисяч. У квітні 2012 року OpenStreetMap очистив 600 000 зареєстрованих учасників. 6 січня 2013 року OpenStreetMap досягла мільйона зареєстрованих користувачів. Близько 30% користувачів внесли хоча б один пункт до бази даних OpenStreetMap.

Наземні обстеження виконуються картографом, пішки, велосипедом, автомобілем, мотоциклом або човном. Дані карти зазвичай збираються за допомогою пристрою GPS, хоча це зовсім не обов'язково, якщо область вже простежена з супутникових зображень.

Після збору даних, вона вноситься до бази даних, завантажуючи її на веб-сайт проекту разом з відповідними атрибутними даними. Оскільки збір і завантаження даних можуть бути відокремлені від редагування об'єктів, внесок у проект можливий без використання пристрою GPS.

Деякі зацікавлені сторони приймають на себе зобов'язання зіставляти цілі міста та міста, або організовувати картування сторін, щоб зібрати підтримку інших, щоб заповнити карту. Велика кількість менш активних користувачів вносять корективи та невеликі доповнення до карти.

На додаток до кількох різних наборів фонових зображень для супутникових зображень, доступних для редакторів OSM, дані з декількох платформ зображення на рівні вулиць доступні як накладання фотографій даних карти: Bing Streetside 360º доріжки зображень, а також відкриті та краудсорсинг платформи Mapillary та OpenStreetCam, як правило, смартфони та інші зображення, встановлені на камері. Крім того, може бути включений шар даних дорожніх знаків Mapillary; це продукт зображень, поданих користувачем.

Примітиви даних OSM зберігаються і обробляються в різних форматах. Основна копія даних OSM зберігається в основній базі даних OSM. Основною базою даних є база даних PostgreSQL з розширенням PostGIS, яка має одну таблицю для кожного примітиву даних, при цьому окремі об'єкти зберігаються як рядки. Усі редагування відбуваються в цій базі даних, і всі інші формати створюються з неї.

Для передачі даних створено кілька звалищ баз даних, які доступні для завантаження. Повне звалище називається planet.osm. Ці звалища існують у двох форматах: один використовує XML і один використовує двійковий формат буфера протоколів (PBF).

Дані LinkedGeoData використовують словники GeoSPARQL і відомі текстові (WKT) RDF для представлення даних OpenStreetMap. Це робота дослідницької групи

Agile Knowledge Engineering і Semantic Web (AKSW) в університеті Лейпцига, групи, яка в основному відома для DBpedia.

Система керування версіями Git

Для можливості розробки в команді та відслідкування усіх гілок розробки та усіх зроблених змін дуже зручно використовувати системи контролю за версіями. В даній роботу використовувалась система Git.

Git - розподілена система керування версіями для відстеження змін вихідного коду під час розробки програмного забезпечення. Він призначений для координації роботи між програмістами, але він може бути використаний для відстеження змін у будь-якому наборі файлів. Його цілі включають швидкість, цілісність даних, підтримку розподілених нелінійних робочих процесів.

Git був створений в 2005 році для розробки ядра Linux, а інші розробники ядра сприяли його початковій розробці.

Як і більшість інших розподілених систем керування версіями, і на відміну від більшості систем клієнт-сервер, кожен каталог Git на кожному комп'ютері є повноцінним сховищем з повною історією та повною функцією відстеження версій, незалежно від доступу до мережі або центрального сервера.

Git є вільним і відкритим програмним забезпеченням, розповсюджуваним за умовами GNU General Public License версії 2.

Примітиви Git по суті не є системою управління вихідним кодом. Багато в чому ви можете просто побачити git як файлову систему - це вміст-адресація, і вона має поняття версій.

З цього початкового підходу до розробки, Git розробив повний набір функцій, що очікуються від традиційного SCM, з особливостями, які в основному створюються в міру необхідності, потім уточнюються і розширюються з плином часу. Деякі потоки даних та рівні зберігання в системі управління перегляду Git.

Git має дві структури даних: змінюваний індекс (який також називається етапом або кешем), який кешує інформацію про робочий каталог і наступну ревізію, яку потрібно ввести; і незмінну, об'єктно-базисну об'єктну базу даних.

Індекс служить точкою підключення між базою даних об'єктів і робочим деревом.

База даних об'єкта містить чотири типи об'єктів: файли (blolb), дерево (tree), фіксації (commit) та теги (tag).

Кожен об'єкт ідентифікується хешем SHA-1 свого вмісту. Git обчислює хеш і використовує це значення для імені об'єкта. Об'єкт поміщається в каталог, що відповідає перших двох символів його хешу. Інша частина хешу використовується як ім'я файлу для цього об'єкта.

Git зберігає кожну ревізію файлу як унікальний блок. Взаємозв'язки між краплями можуть бути знайдені шляхом вивчення дерева і об'єктів фіксації. Нещодавно додані об'єкти зберігаються у всій повноті за допомогою стиснення zlib. Це може швидко споживати великий обсяг дискового простору, тому об'єкти можуть бути об'єднані в пакети, які використовують дельта-стиснення для економії місця, зберігаючи краплі у вигляді своїх змін щодо інших крапель.

Висновки до розділу

У даному розділі були розглянуті основні інструменти розробки для написання веб-інтерфейсу користувача та взаємодію з інтерактивною веб-картою. Було висвітлено їх принцип роботи, порівняно з схожими бібліотеками та фреймворками та обґрунтовано їх використання в даній роботі.

ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В даному розділі описується архітектура програмної реалізації. Спосіб взаємодії з сторонніми сервісами та методика роботи користувача з системою.

Опис архітектури системи

Застосунок буде складатись з веб-сторінки, де користувачеві буде доступна інтерактивна карта та панель обрахунку. Користувач зможе поставити маркер на карті або вибрати вже існуючий. Усі елементи будуть знаходитись на боковій панелі. Такий підхід зробить інтерфейс максимально інтуїтивно зрозумілим (Рисунок 5.1).

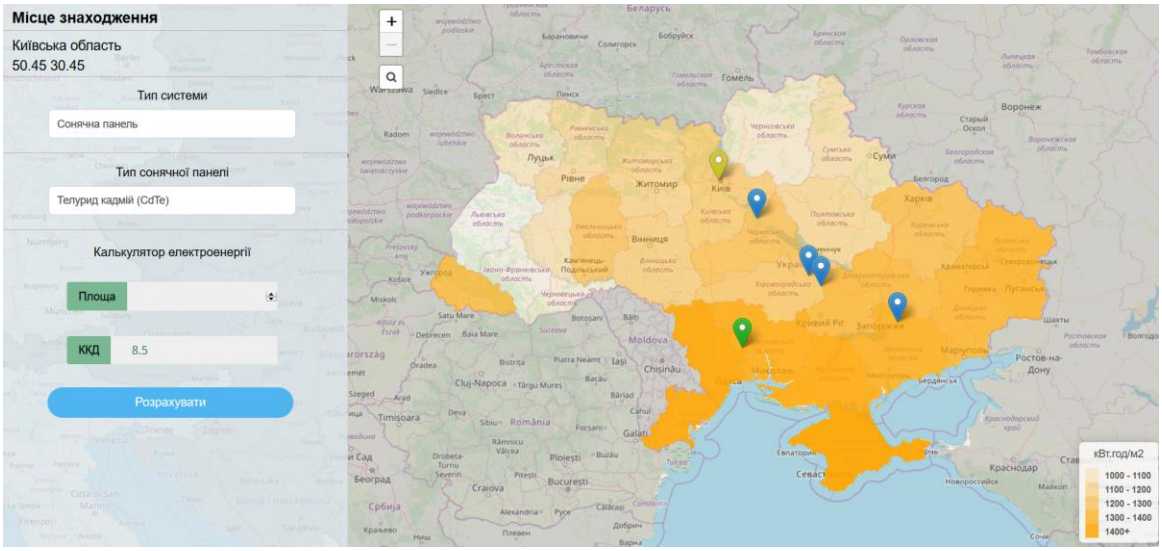

Рисунок 5.1 — Основна сторінка додатку

Система складається з одного актора, який має можливість користуватись веб-картою, аналізувати регіони та типи станцій (Рисунок 5.2).

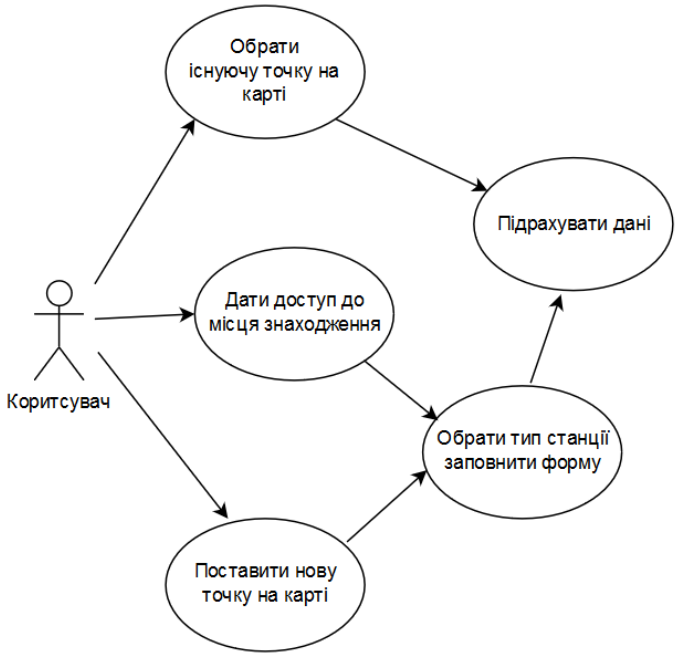

Рисунок 5.2 — Діаграма прецедентів системи

У користувача буде можливість обрати будь-яку точку на карті. Для цього в нього є можливість встановити свій маркер у будь-який регіон України, обрати існуючі маркери, які задає адміністратор системи або дозволити системі самостійно знайти місцеположення користувача.

Користувач може зробити пошук по карті для більш швидкого знаходження потрібного місця. Для цього потрібно ввести в поле пошуку ім’я географічного об’єкту та вибрати потрібний зі списку, який пропонується. Це перенесе користувача к обраному географічному об’єкті, що задано на карті України (Рисунок 5.3).

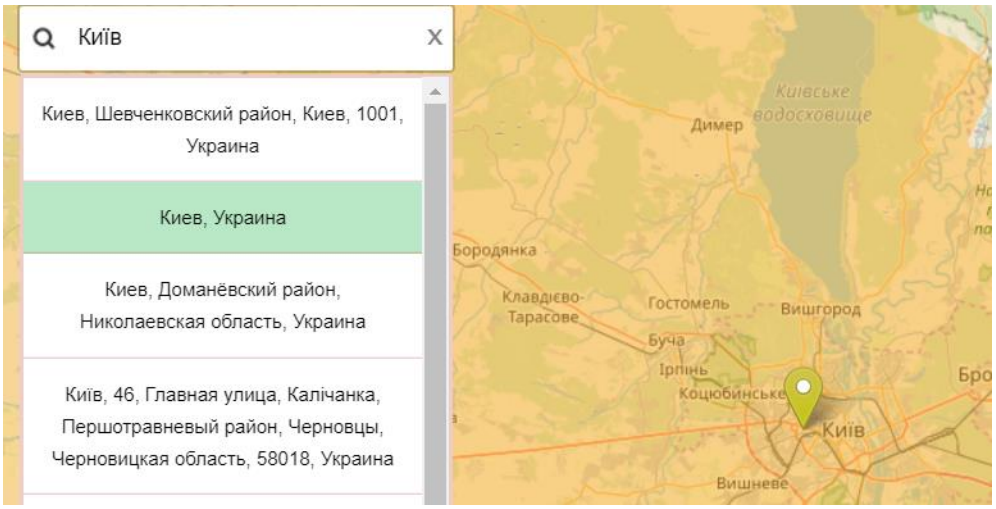

Рисунок 5.3 — Форма пошуку

Користувач має можливість натиснути на існуючу точку на карті. Це автоматично заповнить дані форми в блоці зліва. Маркери блакитного кольору відповідають за сонячні електростанції, маркери зеленого кольору відповідають за геліосистеми для нагріву води. Після натиснення на маркер, користувач може оцінити дані, які притаманні для даної станції та отримати результати (Рисунок 5.4).

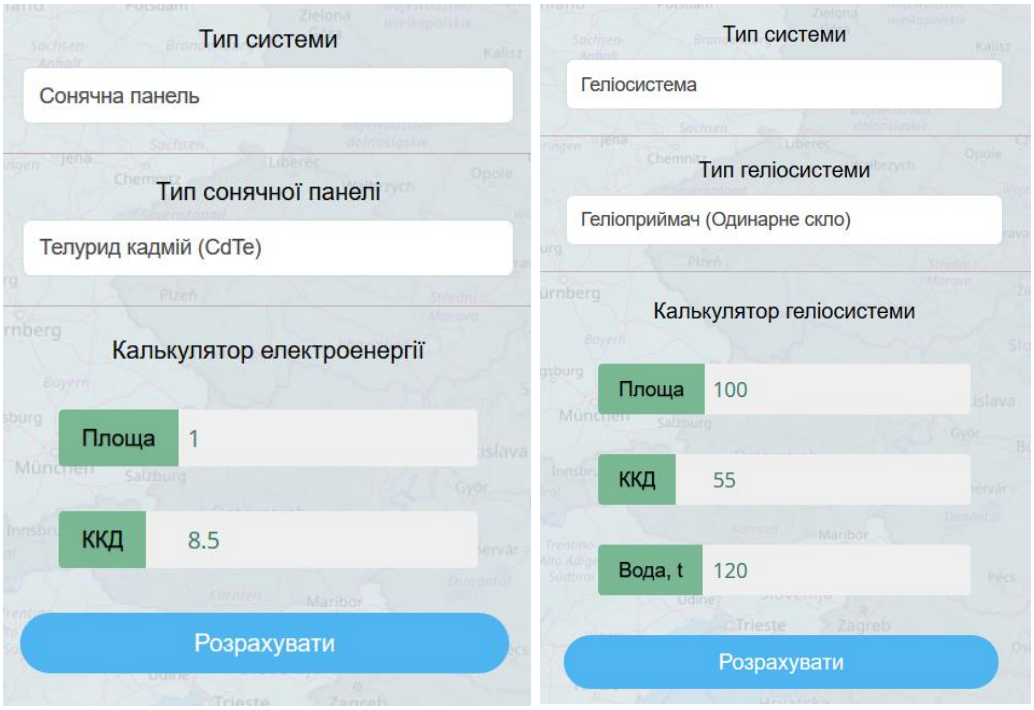

Рисунок 5.4 — Форма обчислення в залежності від маркеру

Також користувач може задати свою точку. Там він повинен обрати тип станції з випадаючого списку (Рисунок 5.5).

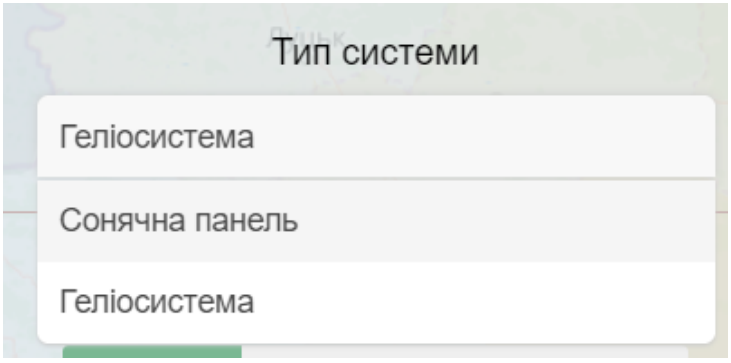

Рисунок 5.5 — Випадаючий список типів станцій

Користувач може обрати конкретний тип сонячної панелі або геліосистеми, від яких залежить к.к.д. та температура вихідної води (Рисунок 5.6, Рисунок 5.7).

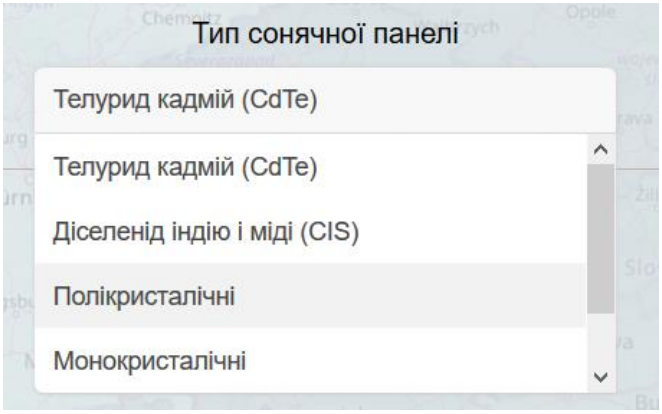

Рисунок 5.6 — Список типів сонячних панелей

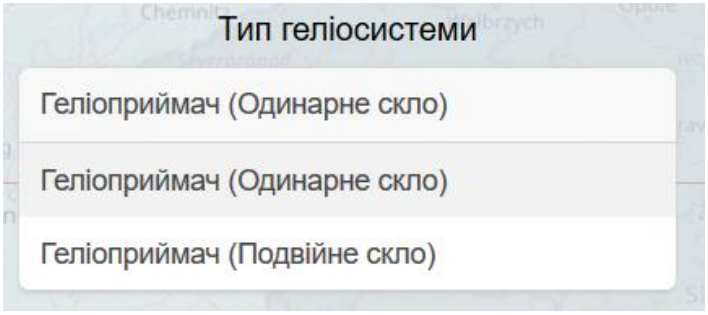

Рисунок 5.7 —Список типів геліостанцій

Після вводу усіх потрібних даних користувач може натиснути на кнопку “Розрахувати” та отримати діаграму в якій вказано помісячний видобуток електроенергії або гарячої води для введених даних та обраного регіону (Рисунок 5.8, рисунок 5.9).

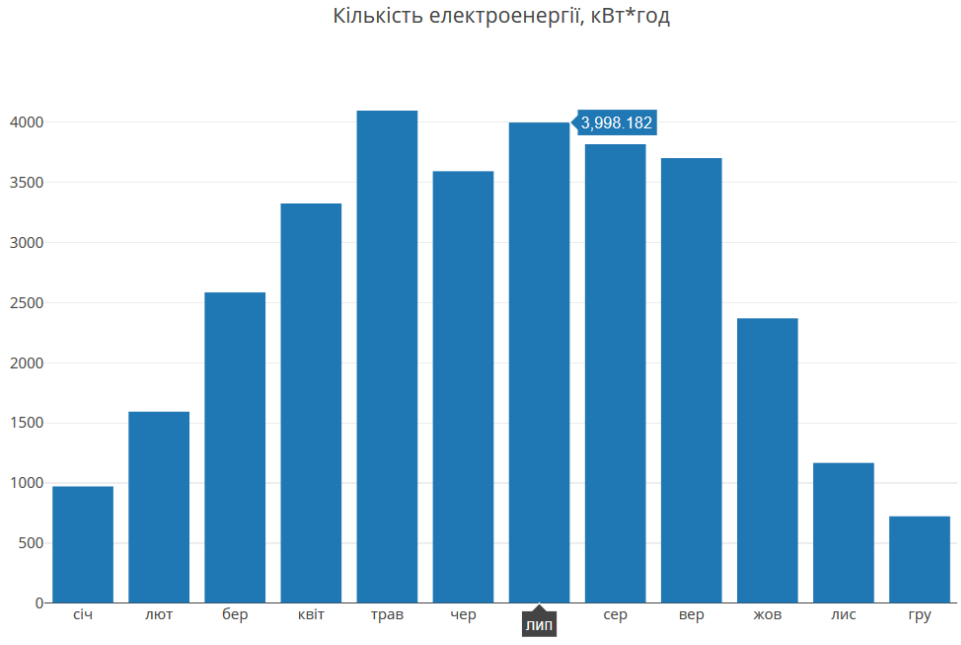

Рисунок 5.8 — Результат обчислень для сонячних панелей у вигляді діаграми

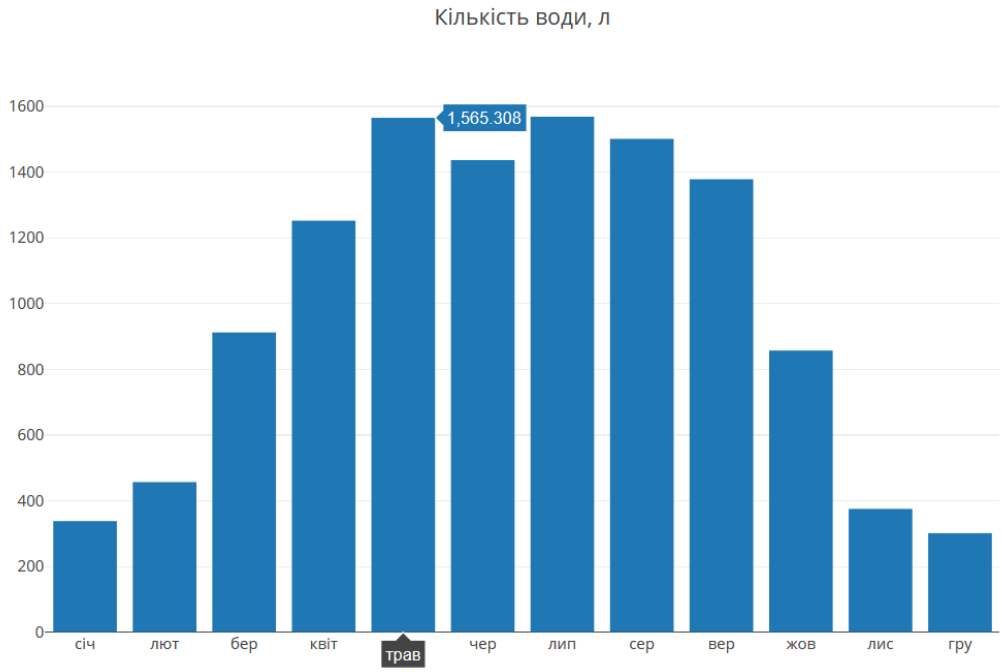

Рисунок 5.9 — Результат обчислень для геліосистем у вигляді діаграми

Розробка сторінки інтерактивної веб-карти

Даний модуль дозволяє користувачу керувати картою та робити обчислення, а саме:

— обирати точки на карті, задані адміністратором;

— обирати самостійно будь-яку точку на карті України;

— обирати тип станції для видобутку електроенергії або гарячої води;

— обирати модель станції;

— перегляд результатів у виді таблиці.

Перш за все було створена файл формату JSON, який відповідає специфікації GeoJson. В ньому вказані координати кожної області України. Імя області визначено за стандартом ISO3166-2.

Наступним кроком було ініціалізація інтерактивної карти за допомогою підключення фреймворку lefleat.js та карти OpenStreetMap. Ім’я областей також відповідає специфікації ІчSO3166-2. Об’єкту карти було передано список областей у GeoJson форматі.

За допомогою іншої компоненти було створено бокову панель, яка виводить місцезнаходження маркера на карті та форми вводу, які реалізовані як окремі компоненти та включені у склад бокової панелі, та обчислення даних в залежності від обраного типу станції у випадаючому списку. Дані з інтерактивної карти попадають на інший компонент за допомогу функції зворотного виклику, яка спрацьовувала при зміні точки на карті.

Реалізація складається з клієнтської та серверної частини. Для їх взаємодії був налагоджено веб-сервіс. Було реалізовано взаємодію з сервером через API(Application Program Interface). Це дає змогу отримання даних щодо сонячної інсоляції по областях України та вже створених точок на карті адміністратором. Ці дані були об’єднані з GeoJson форматом областей та додані до інтерактивної карти.

При ініціалізації карти робиться запит за допомогою API, який повертає дані по сонячній інсоляції по регіонам, типи станцій та встановлені маркери. Після обрання точки на форму можуть бути додані додаткові дані з API, які потрібні для розрахунку (Рисунок 5.10).

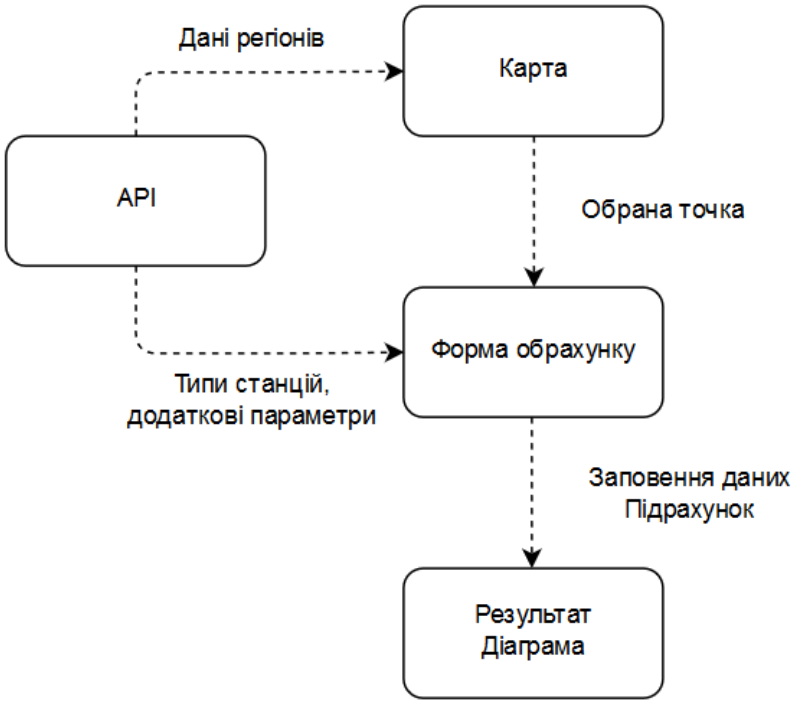

Рисунок 5.10 — Схема структури системи

Головна сторінка надає користувачу усі можливості проекту і складається с декількох компонентів. Компоненти представляють собою класи. Їх взаємодія зроблена так, щоб було нескладно додати нові типи станцій та розробити нову логіки обрахунку даних (Рисунок 5.11).

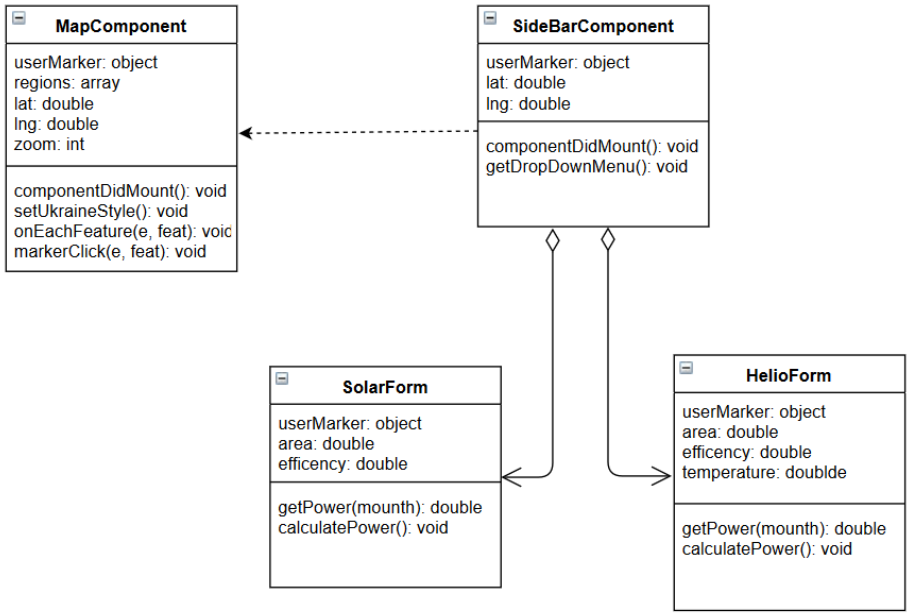

Рисунок 5.11 — Діаграма класів системи

Структура компонентів по факту є деревом, де кожен наступний компонент, якщо потрібно, розбивається на підмножину компонентів (Рисунок 5.12).

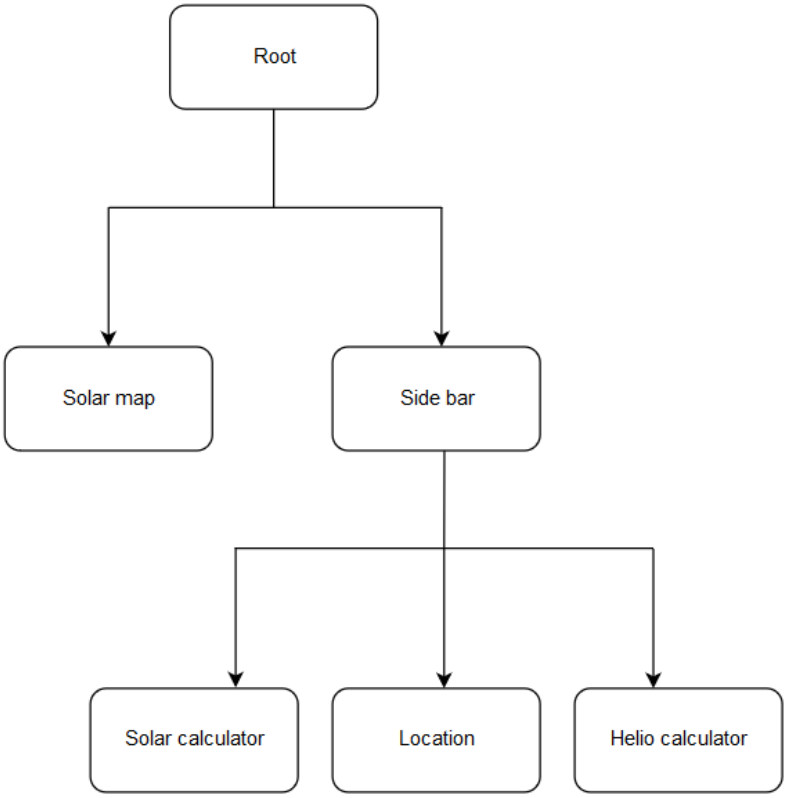

Рисунок 5.12 — Компоненти системи

Висновки до розділу

В даному розділі було розглянуто деталі реалізації системи, спосіб взаємодії між різними сервісами та компонентами та продемонстровано, як користувач повинен працювати з додатком.

ВИСНОВКИ

У ході аналізу існуючого програмного забезпечення для аналізу сонячної активності було досліджено системи, які слугують для вирішення поставленої задачі. Аналіз показав, що існуючі системи вирішують задачу не у повному обсязі.

Розроблений програмний продукт користувачам взаємодіяти з картою онлайн, проводити аналіз. Крім того, користувачі мають змогу користуватись онлайн калькуляторами, експериментувати з показниками в різних регіонах. Модуль написано на мові JavaScript та з використанням React та Leaflet.

Проведено огляд методів і засобів розробки програмної системи. Обґрунтовано вибір створення програмної системи, заснованої на веб-технологіях, а також побудованої за триланковою архітектурою. Це дає змогу підвищити гнучкість та зручність системи, як у розробці та супроводі, так і у використані.

За результатами виконання тестових завдань підтверджена коректність отриманих результатів, отже система відповідає поставленим вимогам. Результати, які отримує користувач, можуть мати похибку (10-20%), яка залежить від особливості місцевості використання сонячних панелей та геліосистем. Дана система істотно спрощує процес моделювання системи видобутку енергії з альтернативних джерел та дозволяє користувачам отримати приблизний результат.

Користувачами системи можуть бути всі, хто бажає використовувати сонячну енергетику або хоче дослідити перспективи. Програмне забезпечення може бути використано на будь-якій операційній системі, на якій встановлено браузер, який підтримує останні веб-стандарти, а також яка має постійний доступ до інтернету.

Також дана система може бути використана не тільки для сонячної енергетики, а може бути розширена для використання інших типів альтернативних джерел енергії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С. О. Кудря. Нетрадиційні та відновлювані джерела енергії / Кудря С. О. − Підручник. – Київ: Національний технічний університет України («КПІ»), 2012.–495с.

2. Н.М.Мхитарян. Энергетика нетрадиционных и возобновляемых источников. К., Наукова думка, 1999. – 314 с.

3. Даффи У.Дж., Бекман У.А. Тепловые процессы с использованием солнечной энергии /Под ред. Ю.Н.Малевского – М., 1977.

4. Твайделл Дж., Уэйр А. Возобновляемые источники энергии. – М.: Энергоатомиздат. 1990. – 344 с.

5. Научно-прикладной справочник по климату СССР, Сер. Вып. 10: УССР. Кн. 1. – Ч. 1. Солнечная радиация и солнечное сияние. – Л.: Гидрометеоиздат, 1990. – 608с.

6. Рекомендации по проектированию установок солнечного горячего водоснабжения для жилых и общественных зданий. Киев, КиевЗНИИЭП, 1987. – 119с.

7. Martin Fowler — GUI Architectures. Часть 1 [Електронний ресурс]. — 2009. — Режим доступу: http://www.rusdoc.ru/articles/18358/

8. Martin Fowler — GUI Architectures. Часть 2 [Електронний ресурс]. — 2009. — Режим доступу: https://habrahabr.ru/post/53536/.

9. Майер Л. — MAVLink Micro Air Vehicle Communication Protocol [Електронний ресурс]. — 2009. — Режим доступу: http://qgroundcontrol.org/mavlink/start.

ДОДАТОК А

Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка веб-інтерфейсу користувача).

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51172_19Б

Аркушів 1

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ _АПЕПС_ ТІ51172_19Б	Записка.docx	Поясню-вальна записка
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ _АПЕПС_ ТІ51172_19Б 12-1	solarMap.jsx solarCalculator.jsx helioCalculator.jsx	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ _АПЕПС_ ТІ51172_19Б 13-1	Додаток В.doc	Опис програмного модуля

ДОДАТОК Б

Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка веб-інтерфейсу користувача).

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51172_19Б 12-1

Аркушів 8

Київ 2019

```
export default class SolarMap extends Component {
  constructor(props) {
    super(props);
    this.state = {
      userMarker: props.userMarker,
      markers: [],
      regions: [],
      angles: [],
      solarStations: [],
      helioStations: [],
      colorStep: 0,
      lat: 48.55,
      lng: 31,
      zoom: 6,
      components: null,
      displayPreloader: true
    };

    this.setUkraineStyle = this.setUkraineStyle.bind(this);
    this.onEachFeature = this.onEachFeature.bind(this);
  }

  componentDidMount() {
    const { withLeaflet } = require("react-leaflet");
    const { ReactLeafletSearch } = require("react-leaflet-search");
    this.setState({
      components: { ReactLeafletSearch: withLeaflet(ReactLeafletSearch) }
    });

    fetch("http://localhost:17725/marks/getAll")
      .then(data => data.json())
      .then(map => {
        this.setState({
          markers: map.markers,
          regions: map.regions,
          angles: map.angles,
          solarStations: map.solarStations,
          helioStations: map.helioStations,
          colorStep: 1.0 / (map.regions.length + 1)
        });

        getLocation().then(location => {
          this.setState({
            userMarker: {
              region: this.state.regions.find(el => el.iso === "UA-32"),
              coordinates: location,
              isNew: true,
              systemStationId: this.state.solarStations[0].id,
              angles: map.angles,
              solarStations: map.solarStations,
              helioStations: map.helioStations
            },
            displayPreloader: false
          });
          this.props.locationCallBack(this.state.userMarker);
        });
      });
  }

  setUkraineStyle(feature) {
    return {
      color: "#FFA500",
      opacity: 0,
      fillOpacity:
        this.state.regions.findIndex(
          el => el.iso === feature.properties["iso3166-2"]
        ) * this.state.colorStep
    };
  }

  onEachFeature(feature, layer) {
    var that = this;
    layer.on("click", function(e) {
      that.markerClick(e, feature);
    });
  }

  markerClick(e, feature) {
```

```
  if (
    e.target &&
    e.target.options &&
    e.target.options.position &&
    e.target.options.position.id
  ) {
    var marker = this.state.markers.find(
      element => element.id === e.target.options.position.id
    );
    var region = this.state.regions.find(
      element => element.id === marker.regionId
    );
    this.setState({
      userMarker: {
        id: marker.id,
        systemStationId: marker.systemStationId,
        area: marker.area,
        region: region,
        regionId: marker.regionId,
        coordinates: e.latlng,
        isNew: false,
        angles: this.state.angles,
        solarStations: this.state.solarStations,
        helioStations: this.state.helioStations
      }
    });
  } else {
    region = this.state.regions.find(
      element => element.iso === feature.properties["iso3166-2"]
    );
    this.setState({
      userMarker: {
        id: null,
        systemStationId: this.state.solarStations[0].id,
        area: 0,
        region: region,
        regionId: region.id,
        coordinates: e.latlng,
        isNew: true,
        angles: this.state.angles,
        solarStations: this.state.solarStations,
        helioStations: this.state.helioStations
      }
    });
  }
  this.props.locationCallBack(this.state.userMarker);

  ReactDOM.unmountComponentAtNode(document.getElementById("results"));
}

render() {
  if (!this.state.components) {
    return null;
  }
  const { ReactLeafletSearch } = this.state.components;
  const yellowIcon = L.icon({
    iconUrl:
      "https://cdn.rawgit.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-yellow.png",
    shadowUrl:
      "https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png",
    iconSize: [25, 41],
    iconAnchor: [12, 41],
    popupAnchor: [1, -34],
    shadowSize: [41, 41]
  });

  const blueIcon = L.icon({
    iconUrl:
      "https://cdn.rawgit.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-blue.png",
    shadowUrl:
      "https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png",
    iconSize: [25, 41],
    iconAnchor: [12, 41],
    popupAnchor: [1, -34],
    shadowSize: [41, 41]
  });

  const greenIcon = L.icon({
```

```
  iconUrl:
    "https://cdn.rawgit.com/pointhi/leaflet-color-markers/master/img/marker-icon-2x-green.png",
  shadowUrl:
    "https://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.7/images/marker-shadow.png",
  iconSize: [25, 41],
  iconAnchor: [12, 41],
  popupAnchor: [1, -34],
  shadowSize: [41, 41]
});

var corner1 = L.latLng(55, 13);
var corner2 = L.latLng(40, 40);
var bounds = L.latLngBounds(corner1, corner2);

let newMark = null;
if (
  this.state.userMarker.coordinates.lat &&
  this.state.userMarker.coordinates.lng
) {
  newMark = (
    <Marker
      position={[
        this.state.userMarker.coordinates.lat,
        this.state.userMarker.coordinates.lng
      ]}
      icon={yellowIcon}
    />
  );
}
var grades = [1000, 1100, 1200, 1300, 1400];
var legendColorStep = 1.0 / (grades.length + 1);

return (
  <React.Fragment>
    {this.state.displayPreloader ? (
      <div class="preloader-wr">
        <div class="lds-ring">
          <div />
          <div />
          <div />
          <div />
        </div>
      </div>
    ) : null}
    <Map
      className="simpleMap"
      center={[this.state.lat, this.state.lng]}
      zoom={this.state.zoom}
      minZoom={this.state.zoom}
      maxBounds={bounds}
      maxBoundsViscosity="0"
    >
      <ReactLeafletSearch
        position="topleft"
        zoom={9}
        showMarker={false}
        showPopup={true}
        openSearchOnLoad={false}
        closeResultsOnClick={false}
        customProvider={false}
        provider="OpenStreetMap"
        providerOptions={{ region: "ua" }}
      />
      <TileLayer url="http://{s}.tile.osm.org/{z}/{x}/{y}.png" />
      <GeoJSON
        data={ukraineGeo}
        style={this.setUkraineStyle}
        onEachFeature={this.onEachFeature}
      />
      <GeoJSON
        data={otherCountriesGeo}
        style={{
          fillColor: "grey",
          weight: 1,
          opacity: 0,
          color: "white",
          dashArray: "3",
          fillOpacity: 0.3
```

```
          }}
        />
        {this.state.markers.map((position, idx) => (
          <Marker
            key={"marker " + idx}
            position={position}
            icon={
              this.state.solarStations.find(
                x => x.id === position.systemStationId
              )
                ? blueIcon
                : greenIcon
            }
            onClick={this.markerClick.bind(this)}
          >
            <Popup className="request-popup">
              <p>{position.text}</p>
              <img width='200px' src={'' + position.imageBase64}/>
            </Popup>
          </Marker>
        ))}

        {newMark}

        <Control position="bottomright" className="legend">
          <div class="legend-header">
            <span>кВт.год/м2</span>
          </div>
          {grades.map((item, i) => {
            return (
              <div>
                <i
                  style={{
                    background: "#FFA500",
                    opacity: legendColorStep * (i + 1)
                  }}
                />
                <span class="legend-text">
                  {item + (grades[i + 1] ? " - " + grades[i + 1] : "+")}
                </span>
                <br />
              </div>
            );
          })}
        </Control>
      </Map>
    </React.Fragment>
  );
 }
}

import React, { Component } from "react";
import ReactDOM from 'react-dom'
import '../../assets/cards.css';
import Plot from 'react-plotly.js';
import $ from 'jquery';

export default class HelioCalculator extends Component {
  constructor(props) {
    super(props);
    this.state = {
      userMarker: props.userMarker
    };
  }

  componentDidMount(){
    const node = ReactDOM.findDOMNode(this);

    if (node instanceof HTMLElement) {
      this.setState({
        dropdown: node.querySelector('#helioDropDown')
      });
    }
  }

  dropDownClick = function(){
    this.state.dropdown.setAttribute('tabindex', 1);
    this.state.dropdown.focus();
```

```
  this.state.dropdown.classList.toggle('active');
  $(this.getDropDownMenu()).slideToggle(300);
 }

 dropDownFocusOut = function(){
  this.state.dropdown.classList.remove('active');
  $(this.getDropDownMenu()).slideUp(300);
 }

 getDropDownMenu = function(){
  const node = ReactDOM.findDOMNode(this);
  if (node instanceof HTMLElement) {
   return node.querySelector('#helioDropDownMenu');
  }
  return null;
 }

 selectStationType = function(item){
  this.props.userMarker.systemStationId = item.id;
  this.setState({
   userMarker: {
    systemStationId: item.id
   }
  });
 }

 handleAreaChange(e) {
  this.props.userMarker.area = e.target.value;
  this.setState({
   userMarker: {
    dropdown: null,
    area: e.target.value
   }
  });
 }

      getPower = function(mounthCount, radiationS, radiationD, lat, efficiency, outWaterTemperature, area, startTemperature) {
         let qS = 0.74;
        let qD = 0.64;
        let closest = this.props.userMarker.angles.reduce(function (prev, curr) {
           return (Math.abs(curr.lat - lat) < Math.abs(prev.lat - lat) ? curr : prev);
        });
        let addAngles = mounthCount >= 6 && mounthCount <= 8 ?
           15 :
           mounthCount >= 10 || mounthCount <= 3 ?
              -15 :
              0;
        let angle = lat + addAngles;
        console.log(this.props.userMarker);
        let pS = this.getMonthValue(this.props.userMarker.angles.find(el => el.lat === closest.lat && el.grade === closest.lat + addAngles), mounthCount);
        let pD = Math.pow(Math.cos(angle * Math.PI / 360), 2);
        return 30 * (efficiency / 100.0) * area * 0.96 * (radiationS * qS * pS + radiationD * qD * pD) / (1.16 * (outWaterTemperature - startTemperature));
      }

plotWithOptions = function(all, elementId) {
 document.getElementById('popup').style.display = 'block';
ReactDOM.render(
<Plot data={all} layout={
 {
  plot_bgcolor: 'rgb(255,255,255, 0.3)',
  width: 900,
  height: 600,
  title: 'Кількість води, л',
  series: {
   stack: 0,
   lines: {
      show: false,
      fill: true,
      steps: false
   },
   bars: {
      show: true,
      align: "center",
      barWidth: 0.8,
      lineWidth: 0,
      fillColor: {
         colors: [{
            opacity: 0.6
```

```
            }, {
                opacity: 0.6
            }]
          }
      },
    },
    grid: {
      hoverable: true,
      borderWidth: 1,
      borderColor: "#979797"
    }
  }
} />, document.getElementById('results'));
}

  render() {
    return (
      <div className='calculator'>
        <div className="select-container" >
          <span className="choose">Тип геліосистеми</span>

          <div id="helioDropDown" className="dropdown" onClick={this.dropDownClick.bind(this)} onBlur={this.dropDownFocusOut.bind(this)}>
            <div className="select">
              <span>{this.props.userMarker.helioStations.find(x => x.id === this.props.userMarker.systemStationId).name}</span>
              <i className="fa fa-chevron-left"></i>
            </div>
            <ul id="helioDropDownMenu" className="dropdown-menu">
              {this.props.userMarker.isNew ? this.props.userMarker.helioStations.map(item => {
                return (<li onClick={this.selectStationType.bind(this, item)}>{item.name}</li>);
              }) : null}
            </ul>
          </div>
        </div>

        <div className='calculator-header'>
          <span>Калькулятор геліосистеми</span>
        </div>

        <div className='calculator-body'>
        <span>
            <input class="basic-slide" type={this.props.userMarker.isNew ? 'number' : ''} min='0' step='0.01' placeholder=""
              value={this.props.userMarker.area}
              onChange={ this.handleAreaChange.bind(this) }
              readOnly={!this.props.userMarker.isNew}/>
            <label>Площа</label>
          </span>
          <span>
            <input class="basic-slide" placeholder=""
              value={this.props.userMarker.helioStations.find(x => x.id === this.props.userMarker.systemStationId).efficiency}
              readOnly/>
            <label>ККД</label>
          </span>
          <span>
            <input class="basic-slide" placeholder=""
              value={this.props.userMarker.helioStations.find(x => x.id === this.props.userMarker.systemStationId).outWaterTemperature}
              readOnly/><label>Вода, t</label>
          </span>
        </div>

        <div className="calculate-buttons">
          <a className="button3" onClick={this.calculatePower.bind(this)}
            disabled={this.props.userMarker.waterOutTemperature}>Розрахувати</a>
        </div>
      </div>
    );
  }
}

export default class SolarCalculator extends Component {
  constructor(props) {
    super(props);
    this.state = {
      dropdown: null,
      userMarker: props.userMarker
    };
  }

  componentDidMount(){
```

```
    const node = ReactDOM.findDOMNode(this);

    if (node instanceof HTMLElement) {
      this.setState({
        dropdown: node.querySelector('#solarDropDown')
      });
    }
  }

  dropDownClick = function(){
    this.state.dropdown.setAttribute('tabindex', 1);
    this.state.dropdown.focus();
    this.state.dropdown.classList.toggle('active');
    $(this.getDropDownMenu()).slideToggle(300);
  }

  dropDownFocusOut = function(){
    this.state.dropdown.classList.remove('active');
    $(this.getDropDownMenu()).slideUp(300);
  }

  getDropDownMenu = function(){
    const node = ReactDOM.findDOMNode(this);
    if (node instanceof HTMLElement) {
      return node.querySelector('#solarDropDownMenu');
    }
    return null;
  }

getPower = function(mounthCount, radiationS, radiationD, lat, efficiency, area) {
  let closest = this.props.userMarker.angles.reduce(function (prev, curr) {
      return (Math.abs(curr.lat - lat) < Math.abs(prev.lat - lat) ? curr : prev);
  });
  let addAngles = mounthCount >= 6 && mounthCount <= 8 ?
      15 :
      mounthCount >= 10 || mounthCount <= 3 ?
          -15 :
          0;
  let angle = lat + addAngles;
  let pS = this.getMonthValue(this.props.userMarker.angles.find(el => el.lat === closest.lat && el.grade === closest.lat + addAngles), mounthCount);
  let pD = Math.pow(Math.cos(angle * Math.PI / 360), 2);
  return (efficiency / 100.0) * area * 0.96 * (radiationS * pS + radiationD * pD);
}

selectStationType = function(item){
  this.props.userMarker.systemStationId = item.id;
  this.setState({
    userMarker: {
      systemStationId: item.id
    }
  });
}

  handleAreaChange(e) {
    this.props.userMarker.area = e.target.value;
    this.setState({
      userMarker: {
        area: e.target.value
      }
    });
  }

plotWithOptions = function(all, elementId) {
  document.getElementById('popup').style.display = 'block';
  ReactDOM.render(
  <Plot data={all} layout={
    {
      width: 900,
      height: 600,
      title: 'Кількість електроенергії, кВт*год',
      series: {
        stack: 0,
        lines: {
            show: false,
            fill: true,
            steps: false
        },
        bars: {
```

```
          show: true,
          align: "center",
          barWidth: 0.8,
          lineWidth: 0,
          fillColor: {
            colors: [{
              opacity: 0.6
            }, {
              opacity: 0.6
            }]
          }
        },
      },
      grid: {
        hoverable: true,
        borderWidth: 1,
        borderColor: "#979797"
      }
    }
  } />, document.getElementById('results'));
}

  render() {
    return (
      <div className='calculator'>
        <div className="select-container" >
          <span className="choose">Тип сонячної панелі</span>

          <div id="solarDropDown" className="dropdown" onClick={this.dropDownClick.bind(this)} onBlur={this.dropDownFocusOut.bind(this)}>
            <div className="select">
              <span>{this.props.userMarker.solarStations.find(x => x.id === this.props.userMarker.systemStationId).name}</span>
              <i className="fa fa-chevron-left"></i>
            </div>
            <ul id="solarDropDownMenu" className="dropdown-menu">
              {this.props.userMarker.isNew ? this.props.userMarker.solarStations.map(item => {
                return (<li onClick={this.selectStationType.bind(this, item)}>{item.name}</li>);
              }) : null}
            </ul>
          </div>
        </div>

        <div className='calculator-header'>
          <span>Калькулятор електроенергії</span>
        </div>

        <div className='calculator-body'>
          <span>
            <input class="basic-slide" type={this.props.userMarker.isNew ? 'number' : ''} min='0' step='0.01' placeholder=""
              value={this.props.userMarker.area}
              onChange={ this.handleAreaChange.bind(this) }
              readOnly={!this.props.userMarker.isNew}/>
            <label>Площа</label>
          </span>
          <span>
            <input class="basic-slide" placeholder=""
              value={this.props.userMarker.solarStations.find(x => x.id === this.props.userMarker.systemStationId).efficiency}
              readOnly/>
            <label>ККД</label>
          </span>
        </div>

        <div className="calculate-buttons">
          <a className="button3" onClick={this.calculatePower.bind(this)}>Розрахувати</a>
        </div>
      </div>
    );
  }
}
```

ДОДАТОК В

Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка веб-інтерфейсу користувача).

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51172_19Б 13-1

Аркушів

Київ 2019

АНОТАЦІЯ

Розділ містить опис частини, яка слугує для роботи з веб-картою, що є основною одиницею програмного продукту, та забезпечує поєднання усії інший компонентів для обрахунку потужності разом у одну веб-сторінку. Вказано принцип роботі з API, який слугує для отримання потрібних даних з серверу та бази даних. Модуль написано мовою програмування C#, з використанням EntityFramework.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку розглядається один з програмних модулів системи — модуль для роботи з веб-сторінкою з кодом УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51172_19Б 12-1, що міститься у файлі solarMap.jsx. Модуль реалізовано за допомогою бібліотек React та Leaflet. Модуль призначений для управління системою, яка відповідають за керування метеоданими, які надходять з API. Користувач має можливість обирати вже існуючі записи або створити новий та розрахувати дані.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Призначенням модулю для роботи з веб-картоє є відображення інформації х серверу та бази даних та безпосереднього прийняття даних введення та обмін інформацією із клієнтським інтерфейсом. Використання такого шаблону дозволяє створювати програмне забезпечення, де інтерфейс і логіка роботи модуля для бази даних та серверу є незалежними компонентами, що дає можливість використовувати його для зменшення навантаження на клієнтську частину системи.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Слідкувати за зміною інформації про підсистеми є головним завданням модуля. При запуску системи модуль повертає всю інформацію, яка міститься в базі даних, для відображення даних на користувацькому інтерфейсі. Також модуль оброблює інформацію, надаючи змогу вводити потрібні дані та проводити обчислення та відображати результати у вигляді діаграми.

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Модуль розроблено у середовищі розробки Microsoft Visual Studio Code, що забезпечує набір сервісних функцій та графічний діалог з користувачем, на комп’ютері, має встановлений браузер та підключення до інтернету. Для реалізації клієнтської частини було використано бібліотеки React та Leaflet. За допомогою елементів React клієнтська частина може з’єднуватись з сервером за допомогою API.

В якості провайдера самої інтерактивної карти було обрано відкритий сервіс OpenStreetMap.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Програмний модуль реалізований як окремий клас, який забезпечує існування клієнтської частини та бізнес-логіки окремо від одного, але разом із цим запуск обох компонентів відбувається одночасно.

Для використання цього модуля треба завантажити веб-сторінку, яка зробить потрібні запити та відобразить результат.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними для модуля є інформація, яку користувач вводить в додатку та місцеположення.

Вихідними даними програмного модуля є результати обчислення, які відображаються у вигляді діаграми.

ДОДАТОК Г

Інтерактивна веб-карта для представлення енергоресурсів та об’єктів відновлюваної енергетики України (розробка веб-інтерфейсу користувача).

Довідки про впровадження результатів роботи

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51172_19Б 14-1

Аркушів 1

Київ 2019